Network/T™

TCPaccess 5.3

User's Guide

COMPUTER ASSOCIATES®
Software superior by design.

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change ore withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON INFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R Sections 52.227-19(c)(1) and (2) or DFARS Sections 252.227.7013(c)(1)(ii) or applicable successor provisions.

© 2000 Computer Associates International, Inc. One Computer Associates Plaza, Islandia, New York 11788-7000. All rights reserved.

All product names referenced herein belong to their respective companies.

Network*IT*™ TCPaccess User's Guide
Table of Contents

Chapter 1 Introduction to TCPaccess

Overview 1-2

Software Components 1-3

- Internet Protocol 1-3
- Internet Control Message Protocol 1-3
- Transmission Control Protocol 1-4
- User Datagram Protocol (UDP) 1-4

Using Host Name Strings 1-5

- Specifying the Host 1-5
- Specifying the Port 1-6
- Specifying Internal VTAM Applications 1-6
- Example Host Name Strings 1-7

Chapter 2 Client FTP2

About File Transfer Protocol 2-3

- Client FTP/Client FTP2 2-3

Client FTP2 2-4

Invoking Client FTP2 2-5

- Invoking Client FTP2 through TSO 2-5
  - FTP2 Command Processor 2-5
  - TSO CALL Command 2-6
- Batch Invocation 2-6
  - Batch Program 2-7
  - Batch TMP 2-7

Client FTP2 Invocation Options 2-8

- Client FTP2 General Options 2-8
  - APP 2-8
  - FIOS 2-8
  - HOST 2-9
  - LOGT 2-9
  - NETRC 2-9

NOA 2-9
NOFIRE 2-9
SYS 2-9
WAIT 2-10
Client FTP2 Debugging Options 2-10
TEST 2-10
TESTI 2-10
VLT 2-10
General Client FTP2 Operation 2-11
Path Name 2-11
Client FTP2 Command Conventions 2-11
Program Prompt 2-11
Completion of Data Transfer 2-12
Case Sensitivity 2-12
Abbreviations in Commands 2-12
Brackets in Commands 2-12
Syntax Conventions 2-12
Example Conventions 2-12
The NETRC File 2-13
Client FTP2 High-Level Qualifier 2-14
Client FTP2 Commands 2-15
? Command 2-17
! Command 2-18
$ Command 2-19
ABOR 2-20
ALLO 2-21
APPEND 2-22
ASCII 2-23
BINARY 2-24
BYE 2-25
CD 2-26
CDUP 2-27
CLOSE 2-28
DEBUG 2-29
DELETE 2-30
DIR 2-31

DISCONNECT 2-32
DO 2-33
EBCDIC 2-34
END 2-35
EXPE 2-36
FIREWALL 2-37
GET 2-39
HELP 2-40
LOG 2-41
LQUOTE 2-42
LS 2-43
MACDEF 2-45
MKDIR 2-47
MODE 2-48
NTRANS 2-49
OPEN 2-50
PUT 2-51
PWD 2-52
QUIT 2-53
QUOTE 2-54
RECV 2-55
REMHELP 2-56
REMSITE 2-57
REMSNDS 2-58
REMSTAT 2-59
RENAME 2-60
REST 2-61
RMDIR 2-63
SEND 2-64
SITE 2-65
SNDS 2-66
STATUS 2-67

STRUCT 2-68
SUNIQUE 2-69
TYPE 2-70
Restart Support 2-72
Using Restart 2-72
Client FTP2 File Transfer Examples 2-73
Example Notes 2-73
PUT Example 2-74
GET Example 2-75
FTP2 Restart Marker Example 2-76
Transfer to an MVS Internal Reader 2-77
Managing Directories on UNIX-based Systems 2-78
Transferring and Using a File in a Single JCL Job 2-79
Examples 2-80

Chapter 3 Client FTP3

Introducing Client FTP3 3-2
PL/I Runtime Libraries 3-2
Connections to the FTP Servers 3-2
Throughput and CPU Utilization 3-2
Client FTP3 Data Transfer 3-2
Client FTP3 3-3
Invoking Client FTP3 3-4
Invoking Client FTP3 through TSO 3-4
FTP3 TSO Command 3-4
TSO CALL Command 3-6
Batch Invocation 3-7
Batch Program 3-7
Batch TMP 3-8
Understanding the Configuration Data Sets 3-9
Dynamic Data Set Allocation 3-10
Related References on the FTP.DATA File 3-12
Changing the High-Level Qualifier 3-12
TCPIP.DATA 3-13
FTP.DATA 3-14
General Client FTP3 Operation 3-17
Path Name 3-17
Client FTP3 Command Conventions 3-18

The NETRC File 3-19
Transferring and Using a File in a Single JCL Job 3-19
Examples 3-20
Client FTP3 Commands 3-22
? Command 3-24
ACCOUNT 3-25
APPEND 3-26
ASCII 3-27
BINARY 3-28
CD 3-29
CDUP 3-30
CLOSE 3-31
DEBUG 3-32
DELETE 3-33
DELIMIT 3-34
DIR 3-35
EBCDIC 3-36
GET 3-37
HELP 3-38
LCD 3-39
LMKDIR 3-40
LOCSITE 3-41
LOCSTAT 3-42
LPWD 3-43
LS 3-44
MDELETE 3-45
MGET 3-46
MKDIR 3-48
MODE 3-49
MPUT 3-51
NOOP 3-52
OPEN 3-53
PASS 3-54

PUT 3-55
PWD 3-56
QUIT 3-57
QUOTE 3-58
RENAME 3-59
RESTART 3-60
RMDIR 3-61
SENDSITE 3-62
SITE 3-63
STATUS 3-64
STRUCT 3-65
SYSTEM 3-66
TRACE 3-67
TSO 3-68
TYPE 3-69
USER 3-71

Chapter 4 Client FTP

Introducing Client FTP 4-3
Client FTP 4-4
Invoking Client FTP 4-5
TSO Invocation 4-5
FTP Command Processor 4-5
TSO CALL Command 4-5
Usage Guidelines 4-6
Invoking FTP as a Batch Program 4-7
Batch Program 4-7
Example 4-7
Batch TMP 4-8
Example 4-8
Client FTP Invocation Options 4-9
Client FTP General Options 4-9
APP 4-9
FIOS 4-9
LOGT 4-10

SYS 4-10
WAIT 4-10
Client FTP Debugging Options 4-10
DISP 4-10
TEST 4-11
TESTI 4-11
VLT 4-11
General Client FTP Operation 4-12
Path Name 4-12
Client FTP Command Conventions 4-12
Example 4-13
The Host Prefix 4-13
Client FTP Commands 4-14
? Command 4-16
ABOR 4-17
ADD 4-18
ALLO 4-19
BYE 4-20
CDUP 4-21
CONN 4-22
CWD 4-23
DELE 4-24
DO 4-25
END 4-26
EXPE 4-27
GET 4-28
HELP 4-29
LIST 4-30
LOG 4-31
MKD 4-32
MODE 4-33
NLST 4-35
PUT 4-36
PWD 4-37

QUIT 4-38
QUOT 4-39
REN 4-40
REST 4-41
RMD 4-43
SEND 4-44
SITE 4-45
SNDS 4-46
STAT 4-47
STRU 4-48
TYPE 4-49
A=B 4-51
A?B 4-52
Restart Support 4-53
How to Restart 4-53
Client FTP File Transfer Examples 4-54
FTP Invocation and Conventions 4-54
Host Prefixes 4-54
Successful Completion of a Transfer 4-54
Entering Text 4-54
Readability 4-55
PUT Example 4-56
GET Example 4-57
Transferring and Using a File in a Single JCL Job 4-58
Examples 4-58
Transfer to an MVS Internal Reader 4-59
Restart File Transfer Example 4-60
Managing Directories on UNIX-based Systems 4-63

Chapter 5 Server FTP

Introducing Server FTP 5-2
File Handling by the Server FTP 5-3
Handling a Record Which is Too Long 5-3
Transferring Files to a Host 5-3
Sophisticated File Handling 5-4
Transferring Files To a Tape 5-4

Configuration 5-4
SITE Command Parameters 5-5
Using FTP to Write to Magnetic Tape 5-5
Server FTP Commands 5-7
ALLO 5-9
HELP 5-10
MKD 5-12
REST 5-13
RMD 5-14
SITE 5-15
Site Command Parameters 5-17
STAT 5-27
Data Set Attributes 5-28
Units and Volumes 5-28
Data Set Names 5-28
FTP Path Name Syntax 5-29
Using Wildcard Characters in FTP 5-30
Partitioned Data Sets 5-31
VTOC and Catalog 5-32
MVS Space Allocation 5-32
Multivolume Data Sets 5-33
Data Set Organization 5-33
Disk Format (DCB) Attributes 5-33
Sequential Data Sets 5-34
Partitioned Data Sets 5-34
Default Data Set Attributes 5-34
Generic Attribute Names 5-35
Rules for Record Formats 5-36
Data Set Attribute Errors 5-36
Appending to Empty Data Sets 5-36
JES Internal Reader Support Procedures 5-37
Data Transfer Operations 5-39
Transfer Commands 5-39
Truncating and Folding Records 5-39
Raveled Files 5-39
Padding Fixed-Length Records 5-40
Translation 5-40
Line Image Files 5-40

Record Structured Files 5-40
Tabs 5-40
Carriage Control and Format Effectors 5-41
Character Type Rules 5-42
File Structure with No Format Control 5-42
Line Image Files 5-42
Storing Line Image Files 5-42
Retrieving Line Image Files 5-43
Raveled Files 5-43
Storing Raveled Files 5-43
Retrieving Raveled Files 5-44
File Structure with Telnet Format 5-44
File Structure with ASA Format 5-44
Line Image Files 5-45
Storing Line Image Files 5-45
Retrieving Line Image Files 5-45
Raveled Files 5-45
Storing Raveled Files 5-46
Retrieving Raveled Files 5-46
Record Structure with No Format 5-46
Storing Logical Records 5-46
Retrieving Logical Records 5-47
Record Structure with ASA Format 5-47
Storing Print Files 5-48
Retrieving Print Files 5-48
Binary-Type Rules 5-49
No Record Structure 5-49
Storing Binary Files 5-49
Retrieving Binary Files 5-49
Record Structure 5-50
Storing Structured Binary Files 5-50
Retrieving Structured Binary Files 5-50
Non-Invertible Retrieval 5-51
Sources of Non-Invertibility 5-51
Other Features. 5-52
Testing and Debugging 5-52
Telnet Break 5-52

Chapter 6 Client/Server Telnet

Introducing Client/Server Telnet 6-2

Client Telnet. 6-3

The TSO Client Telnet Command 6-3

The TELNET Command 6-4

General Command Options 6-4

Debug Options 6-5

TSO Client Telnet Operation 6-5

Line Mode Operation 6-5

Screen Mode Operation 6-6

The NULL Transaction 6-7

Sending Data 6-7

Command Entry Rules 6-8

Retransmitting Data 6-8

Function Keys 6-8

TSO Client Telnet Commands 6-9

Commands for Sending Data 6-9

Commands for Session Control 6-10

Commands for Controlling Input and Output 6-10

Miscellaneous Commands 6-13

VTAM Client Telnet 6-14

Invoking VTAM Client Telnet 6-14

VTAM Client Telnet Operation 6-14

NVT Operation from 3278 Terminals 6-15

Full-Screen Operation from 3278 Terminals 6-16

Client Telnet for ASCII Terminals 6-16

Invoking Client Telnet from an ASCII Terminal 6-17

Server Telnet 6-18

Server Telnet Commands 6-18

Autologon to Specific VTAM Applications 6-19

USS Table Support for Server Telnet 6-19

Telnet Escape Sequences. 6-20

Valid Escape Sequences 6-20

Chapter 7 Simple Mail Transfer Protocol

Introducing SMTP 7-2

Mail Transport Support Programs 7-2

Understanding TCPaccess Mail Services 7-3

Mail Service Components 7-4

Using the Mail Facilities 7-5
Getting Started 7-5
Receiving Mail 7-5
Sending Mail 7-6
Invoking SNDMSG 7-6
SNDMSG Session 7-6
SNDMSG Termination 7-7
Interface to a User Mail System 7-8
Receiving Mail 7-8
SSMTP 7-8
Sending Mail 7-9
SPOOL#4 7-9
Sample Jobs 7-10
USMTP 7-11

Chapter 8 USPOOL

Overview of the Print Server Program 8-2
UNIX Print Facility 8-3
The UNIX lp Commands 8-3
lpr 8-3
lpq 8-3
lprm 8-3
Line Printer Daemon 8-3
Remote Printer Definition 8-3
Protocol and Format 8-4
Overview of the lpr Protocol 8-4
Data File Format 8-6
Control File Format 8-6
Print Server Facility 8-8
USPOOL Overview 8-8
ASCII-EBCDIC Translation 8-8
Control Character Translation 8-8
Backspacing 8-8
Horizontal Tab 8-8
Line Feed 8-9
Form Feed 8-9
Carriage Return 8-9
Limitations and Restrictions 8-9
USPOOL Configuration 8-9

Chapter 9 Remote Executor

What Is the Remote Executor? 9-2

The REMCMND Program 9-2

Running REMCMND Interactively (TCPREXEC) 9-3

- Using the NETRC File with TCPREXEC 9-4

Running REMCMND In Batch Mode 9-5

- Using a NETRC file in Batch Mode 9-5
- Command Line Arguments 9-6
- Combining Commands 9-6
- Using a SYSIN DD Statement In Your JCL 9-7
  - Using the \ Character for Continuing Long Lines 9-7

Chapter 10 Using TCPaccess API and Socket Applications

What Example Applications are Provided? 10-2

ACSHELLO 10-3

- Invoking ACSHELLO 10-3

FINGER 10-4

TTCP 10-5

WHOIS. 10-8

Chapter 11 Programmable Batch Interface For FTP and FTP2

What Is the Programmable Batch Interface? 11-2

FTPBATCH Interfaces 11-3

- Batch Control Fields 11-3
  - Command Field 11-3
  - Length 11-4
  - Return Code 11-5
  - Control1 11-6
  - Control2 11-6
  - Secondary LU (Send) 11-6
  - Secondary LU (Receive) 11-6
  - Response 11-7
  - Data Control Field 11-7

Coding The Batch Application 11-8

- Authorized Program Facility (APF) 11-8
- Application Selection (SYSOPT) and Invocation Options 11-9
- User Program Batch Invocation 11-9

Execution Samples 11-10
Sample Using FTP 11-10
Sample Using FTP2 11-10
Batch Program 11-10
TMP Execution 11-11
Debugging Information 11-12
Environment 11-12
PL/I Environment 11-12
FTP and FTP2 Debugging Options 11-12
Message Output 11-12
Sample Batch Programs 11-13
COBOL Language Batch Application 11-13
Assembler Language Batch Application 11-13
Sample Batch Application Linkedit 11-13
Sample Batch Application SYSPUT Output 11-13

Index

1

Introduction to TCPaccess

The ***TCPaccess User's Guide*** provides users with guidelines for using communications programs on Multiple Virtual System (MVS) operating systems on which TCPaccess is installed.

This guide describes command and program usage instructions for users working on an MVS system running TCPaccess. Users who need to send mail, use Telnet to log on to remote computers, or use File Transfer Protocol (FTP) to transfer files among connected computers will find instructions for the necessary commands and their options in this guide.

This chapter describes TCPaccess, its capabilities, and the programs it provides. In it, you will find these chapters:

- "Overview" on page 1-2

  Describes TCPaccess, its functions within a network, and the role TCPaccess plays in your network communications.

- "Software Components" on page 1-3

  Describes the TCPaccess internal components and how they provide their services to users.

- "Using Host Name Strings" on page 1-5

  Describes how to make connections to hosts.

Overview

TCPaccess is a communication subsystem for the Department of Defense internet protocols. It runs on an IBM 370-style mainframe under the MVS operating system.

TCPaccess includes this Internet-specific protocol code:

- The local network I/O driver
- Code for the host-to-host protocol TCP/IP
- Programs for the user-level protocols

TCPaccess supports these standard user-level protocols:

- Telnet for remote login
- File Transfer Protocol (FTP) for file transfer
- Simple Mail Transfer Protocol (SMTP) for electronic mail

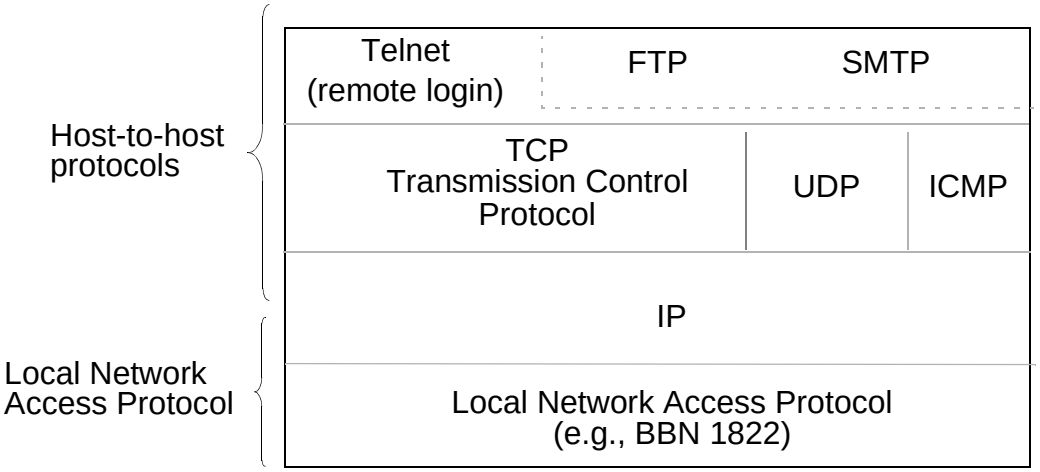
You can install TCPaccess on MVS/ESA with no operating system modifications; inter-process communication is processed with IBM's ACF/VTAM and Cross-Memory Services.

Software Components

The TCPaccess software provides several different protocols. (See “Figure 1-1” on page 1-3).

- Internet Protocol (IP)
- Internet Control Message Protocol (ICMP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- The user level protocol, including Telnet, File Transfer Protocol (FTP), and Simple Mail Transfer Protocol (SMTP)

Figure 1-1 Protocol Hierarchy within TCPaccess



Internet Protocol

IP provides datagram service in an inter-network environment. It sends datagrams between hosts that may be on different networks linked by packet-switching hosts called gateways. A datagram consists of an IP header followed by data.

From a host viewpoint, IP provides the principal functions of host addressing, fragmentation, and reassembly of packets, enabling communication between networks with different packet sizes. IP does not provide error detection and recovery.

Internet Control Message Protocol

ICMP is an extension of IP and carries routing, congestion control, and error reports to hosts.

Transmission Control Protocol

Transmission Control Protocol (TCP) is a reliable end-to-end protocol for transmitting data between processes over connections or virtual circuits. TCP uses IP to carry data packets and is the transport service protocol that most user-level protocols use. These are some characteristics of TCP:

- Segments

  TCP sends data in messages called segments, each of which begins with a TCP header and is sent as a datagram using IP. TCP delivers data segments to a user reliably and in order. The data in these ordered segments logically form an uncommitted stream of 8-bit data bytes or octets.

- Flow Control

  TCP provides flow control on each connection using a windowing mechanism in a fine-grain sequence space — in other words, a single octet of the data stream.

- Checksums

  TCP uses checksums to ensure end-to-end reliability. The receiver sends acknowledgments of correctly received data, and the sender does timer-based retransmission of unacknowledged data.

- Full-duplex Connections

  TCP creates full-duplex connections whose ends are labeled with 16-bit numbers called ports. A TCP connection is defined by a set of four address parameters:

(*local_host_IP_address,local_port remote_host_IP_address,remote_port* **)**

  TCP allows the same local port on a host to participate in any number of connections whose remote ends have differing pairs (remote_host_IP_address, remote_port). Thus, a server host's well-known port (wkport) can participate in multiple TCP connections as long as the user host's pairs (remote_host_IP_address, remote_port) are each unique.

- Connection Management

  A TCP connection is full-duplex, even if an application needs only a simple connection. Furthermore, a TCP connection is allowed to be half-open indefinitely. A close request (FIN) signals the end of data transmission in only one direction; data flow can continue in the other direction until a matching FIN is sent. The connection is fully closed and deleted only when both ends send close requests.

User Datagram Protocol (UDP)

UDP provides datagram service between two processes. UDP does not define connections as TCP does. UDP does, however, have 16-bit port numbers like TCP; sending or receiving a UDP datagram requires the same set of four address parameters that define a TCP connection. As a result, UDP implementation is much like that of TCP, except UDP is simpler.

Using Host Name Strings

Both User Telnet and User FTP programs require you to identify the remote host to which a connection is to be established. Depending on the service to which the connection is established, you might also want to specify optional parameters, such as port number.

Host name strings comprise three sections as illustrated in the following:

host<route>,port

The only required portion of the host string is the host section.

Specifying the Host

Each host is assigned an Internet host number, or Internet address. Because Internet numbers are hard to remember and keep track of these host numbers, a Domain Name System tracks Internet addresses and correlates them to host names. So you can use names instead of numbers to reference host computers.

host_name.network_name.

Example

In the following example, the name identifies a particular host, UNIX, (host number 123.196.222.160) in the COMPANY.NAME.COM system. Notice that the name is terminated with a period. This identifies the name as being fully qualified.

UNIX.*COMPANY***.***NAME***.COM.**

When you enter a fully qualified host string, that is, a host string with the trailing period, TCPaccess processes it exactly as it was entered. When the name is not fully qualified, TCPaccess tries to resolve it using a search list set up by the system administrator. If the search list is properly set up, a host string such as HOST1 can be entered and TCPaccess processes it as if HOST1.COMPANY.COM. had been entered.

Sometimes a host name has an alternate name, or an alias, defined for it. If the target host has an alias, you can enter the alias. HOST1.COMPANY.COM. has the alias UNIX. Entering either of these host strings causes a connection to host 123.196.222.160.

As an option, you can enter the Internet host number instead of the host name. Internet host numbers consist of four integers (between 0 and 255) separated by periods. This is known as dotted decimal notation. Entering 26.0.0.73 specifies the host name NIC.DDN.MIL.

Note: The host parameter of the host string is required.

Specifying the Port

The port option lets you select the port number on the destination host. By default, Telnet connects to port 23 and FTP connects to port 21. If a port variable is used, it must have a leading comma to separate it from the host or *<route>* options.

The port number is an integer with a value between 1 and 65535. If the port number is greater than 999, do not use a comma within the port number. For example, enter 1023, not 1,023.

Note: The port option of the host string is optional.

Example

The following command connects to port 1023 at host hostA.

hostA,1023

Specifying Internal VTAM Applications

In addition to connecting to remote hosts through TCP/IP, you can connect to applications internal to TCPaccess. These applications are:

- **VTAMTEST**

  Invokes **ACTEST**, but instead of communicating over TCP/IP, the communication is through VTAM.

- **SNDMSG**

  Allows a user to send mail messages to other internet users.

- **STELWHO**

  Invokes a **WHOIS** client function.

- **BCAST**

  Invokes a broadcast facility to send messages to all or selective TCPaccess users.

Examples

Specify internal VTAM applications by entering a semi-colon (**;**) followed by the name of the application, as in the following example:

;SNDMSG

Some internal applications, when invoked, can be passed optional parameters. The first blank in the optional parameter field terminates the parameter. Optional parameters can follow the application name and are separated from it by a slash (***/***), as in the following example:

;stelwho/nic

Example Host Name Strings

The following example host name strings all perform the same function; each string specifies a connection to the Telnet Server at host NIC.DDN.MIL., which has an ARPANET address (host number) of 10.0.0.51:

nic.ddn.mil.	Fully qualified name
sri-nic.arpa	Alias of NIC.DDN.MIL.
10.0.0.51	Host number NIC.DDN.MIL
nic	Only if ddn.mil. is in search list.

The following examples are more complex:

nic.ddn.mil.,9	Telnet connection to port 9 (discard)
128.16.9.3<10.0.0.51>,17	Suggest gateway route; use port 17

The following are examples for using internal applications:

;sndmsg	Invoke Send Mail
;vtamtest	Invoke ACTEST using VTAM interface
;stelwho/nic	Invoke WHOIS client
;bcast	Invoke Broadcast function

2

Client FTP2

This chapter describes Client FTP2, the File Transfer Protocol (FTP) that allows file transfers among unlike hosts in diverse internetworking environments. It contains these sections:

- "About File Transfer Protocol" on page 2-3

  Provides a brief overview of the TCPaccess File Transfer Protocol implementation.

- "Client FTP2" on page 2-4

  Provides a flow chart of how FTP works.

- "Invoking Client FTP2" on page 2-5

  Describes how to use Client FTP2 as both a TSO command processor and as a regular batch program.

- "Client FTP2 Invocation Options" on page 2-8

  Describes both the general and debug options available with the FTP2 command.

- "General Client FTP2 Operation" on page 2-11

  Describes the general operation of the Client FTP2 program.

- "Client FTP2 Commands" on page 2-15

  Describes each of the Client FTP2 commands and includes a table listing all the commands with a brief description of each.

  Commands are:

"? Command" on page 2-17	"! Command" on page 2-18	"$ Command" on page 2-19	"ABOR" on page 2-20	"ALLO" on page 2-21
"APPEND" on page 2-22	"ASCII" on page 2-23	"BINARY" on page 2-24	"BYE" on page 2-25	"CD" on page 2-26
"CDUP" on page 2-27	"CLOSE" on page 2-28	"DEBUG" on page 2-29	"DELETE" on page 2-30	"DIR" on page 2-31
"DISCONNECT" on page 2-32	"DO" on page 2-33	"EBCDIC" on page 2-34	"END" on page 2-35	"EXPE" on page 2-36
"FIREWALL" on page 2-37	"GET" on page 2-39	"HELP" on page 2-40	"LOG" on page 2-41	"LQUOTE" on page 2-42

“LS” on page 2-43	“MACDEF” on page 2-45	“MKDIR” on page 2-47	“MODE” on page 2-48	“NTRANS” on page 2-49
“OPEN” on page 2-50	“PUT” on page 2-51	“PWD” on page 2-52	“QUIT” on page 2-53	“QUOTE” on page 2-54
“RECV” on page 2-55	“REMHELP” on page 2-56	“REMSITE” on page 2-57	“REMSNDS” on page 2-58	“REMSTAT” on page 2-59
“RENAME” on page 2-60	“REST” on page 2-61	“RMDIR” on page 2-63	“SEND” on page 2-64	“SITE” on page 2-65
“SNDS” on page 2-66	“STATUS” on page 2-67	“STRUCT” on page 2-68	“SUNIQUE” on page 2-69	“TYPE” on page 2-70

- “Restart Support” on page 2-72

  Describes how to restart an interrupted file transfer.

- “Client FTP2 File Transfer Examples” on page 2-73

  Provides examples of some of the Client FTP2 commands.

About File Transfer Protocol

The File Transfer Protocol (FTP) is an application protocol in the Internet protocol suite. It allows file transfers among unlike hosts in diverse internetworking environments. Using FTP, you can move a file from one computer to another, even if each computer has a different operating system and file storage format. Files can contain data, programs, text, or anything that can be stored online.

The objectives of the FTP protocol are to

- provide sharing of files (computer programs and/or data)
- encourage indirect or implicit use of remote computers through programs
- shield users from variations in file storage systems among hosts
- transfer data reliably and efficiently

FTP is based on a model of files having a few attributes and a mechanism for processing commands and replies. The command-and-reply mechanism establishes the parameters for a file transfer, then performs the transfer. Like Telnet, FTP runs over TCP and assumes the service level provided by TCP.

These documents define FTP:

- *RFC 959, File Transfer Protocol (FTP)*
- *MIL-STD-1780, Military Standard File Transfer Protocol*

Client FTP/Client FTP2

Two versions of Client FTP are provided with TCPaccess: Client FTP2 and Client FTP.

Client FTP2 operates as a three-party model in which there are two control connections and one data connection. The FTP2 control connection and the local host connection are one and the same.

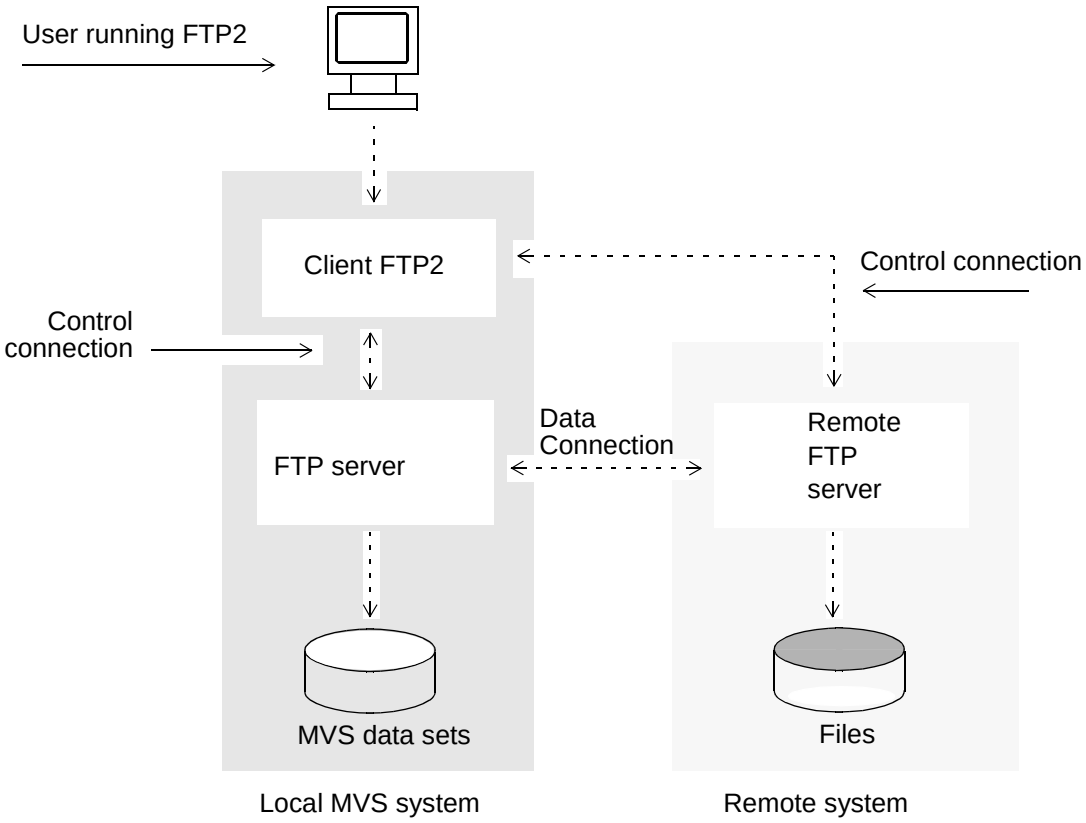
The FTP protocols require that you connect to a host before most commands can be issued. With the Client FTP2 program, an MVS TSO or batch user signs on to the remote host only. Client FTP2 automatically connects and logs on to the TCPaccess MVS host as the local host (thus hiding one side of the three-party model). So after a single sign-on to the remote host, the user can transfer files between the local host and the remote host.

Although Client FTP and Client FTP2 both operate internally as third-party models, Client FTP2 appears to you, the user, to be a two-party model. Also, Client FTP2 commands are designed to resemble UNIX FTP commands; therefore, Client FTP and Client FTP2 commands differ somewhat.

Client FTP2

"Figure 2-1" on page 2-4 shows the relationship between the Client FTP2 program and two FTP servers in the ***three-party model***, indicating that three connections are maintained. Two connections are for command control and one is for data flow.

Figure 2-1 Client FTP2 program in the Three-Party Model



In "Figure 2-1" on page 2-4, the user is logged in to the local MVS system and is running FTP2 under TSO. Through FTP2, the user opens a connection to the remote FTP server. The connection to the local FTP server is opened automatically by the Client FTP2 program and is hidden from the user. FTP2 maintains two control connections: One for communication with the TCPaccess FTP server and the second with the remote FTP server.

With FTP2, the user has access to files on both systems. Data transfer occurs directly between the local and remote servers. Data does not pass through the Client FTP2 application.

Invoking Client FTP2

The Client FTP2 program runs as both a TSO command processor and as a regular batch program. This means that you can use Client FTP2 as a TSO command or call it as a regular program with MVS JCL.

Invoking Client FTP2 through TSO

In a TSO environment, Client FTP2 can be accessed as a TSO command processor or can be called as a program with the TSO **CALL** command. Because Client FTP2 does not use full-screen facilities, it can be used from any type of terminal supported by TSO, including 3270 systems, 3767 systems, and asynchronous ASCII terminals supported by NTO or NPSI.

Note: You must have PROMPT set in your TSO profile for FTP2 to work properly in interactive mode.

FTP2 Command Processor

Invoke Client FTP2 by entering the **FTP2** TSO command in this format.

FTP2 [*/ option1 option2 ..*]

Syntax Description

ftp2	Invokes the Client FTP2 program.
/options	Any number of FTP2 invocation options can be specified in the invocation command.

Client FTP2 responds with either of these messages:

```
Client FTP2 Enter command or '?'
```

or

```
Setting local directory to the TSO profile prefix
250 "USER1." is current prefix
Client FTP2 Enter command or '?'
```

At this point, you are automatically connected and logged onto the MVS host. The working directory on the local host is set to the TSO profile prefix. If no prefix is defined, the working directory is set to the TSO user ID.

FTP2 invocation options are described in "Client FTP2 Invocation Options" on page 2-8.

Usage Guidelines

All Client FTP2 options must be preceded by a slash (/). If the slash is omitted, the options are not recognized by Client FTP2.

TSO CALL Command

Use the tso **call** command in a TSO environment to call and execute the FTP2 program out of a specific library. This is especially useful at sites that run multiple releases of the product or have test and production versions of the product at different maintenance levels.

CALL 't01tcp.link(ftp2)' ['/option1 option2 ..']

Syntax Description

't01tcp.link(ftp2)'	Version of FTP2 being used.
'/options'	Any number of FTP2 invocation options can be included in the **call** command.

Usage Guidelines

- The data set name, t01tcp.link, might need to be replaced by the appropriate data set name at your installation. Check with your TCPaccess site administrator.
- When options are specified in the command statement, they must be enclosed in single quotes.
- When invoked by the **call** command, Client FTP2 runs as a program and not as a TSO command processor. It is not necessary to allocate data sets before calling FTP2 with the **call** command.
- Under TSO, any data set needed by the Client FTP2 program is dynamically allocated and freed.

See "Client FTP2 Invocation Options" on page 2-8 for detailed descriptions of the FTP2 invocation options.

Batch Invocation

The Client FTP2 program can be run in batch as either a program like any other, or as a TSO command processor by running it under a batch Terminal Monitor Program (TMP). FTP2 commands **DO**, **DIR**, and **LS** require a TSO environment to execute, thus they only execute in batch under the TMP.

Note: When running in batch mode, specify the Client FTP2 commands carefully, because the slightest error can cause all subsequent commands to fail and force you to rerun the batch job.

For PL/I V2.2.1 users, FMID PLIX150 must be installed. The TCPaccess LINK library must be placed before PL/I runtime libraries in your STEPLIB concatenation for client batch jobs (including FTP, FTP2, ACPEEP, or TELNET). Failure to do this may cause IBM002I, IBM004I, or IBM014I messages.

When run in batch, Client FTP2 sets a program condition code depending on the severity of Client FTP2 errors encountered; a return code of 0 indicates that all commands entered were successful.

You can specify a NETRC file in batch mode. Specify a NETRC DD file with the name of your NETRC file. Use the NETRC invocation option described in "Client FTP2 Invocation Options" on page 2-8.

Batch Program

You can invoke Client FTP2 in batch in a manner similar to any other batch utility program. Invoking FTP2 as a batch program does not provide a TSO environment that the **DO**, **DIR**, and **LS** commands need. Thus these commands are rejected when FTP2 executes as a batch program.

```
//jobname    JOB job_stmt_parms
//FTPSTEP    EXEC PGM=FTP2,PARM='/ option1 option2...'
//STEPLIB    DD DSN=T01TCP.LINK,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSPUT     DD SYSOUT=*,DCB=BLKSIZE=133
//SYSGET     DD *,DCB=BLKSIZE=80
open unix
user pass
mkdir /u/lpn/d.new
```

Optionally, you can include a NETRC file, as shown here:

```
//NETRC DD DISP=SHR,DSN=userid.FTP.NETRC
```

Read “Client FTP2 General Options” on page 2-8 for detailed descriptions of the FTP2 options.

Batch TMP

You can invoke Client FTP2 in batch as a TSO command processor by running it under a batch TMP.

In the following example, the batch TMP program is IKJEFT01, which is the normal TSO TMP. Since the batch TMP provides the TSO environment, all FTP2 commands are available.

For this example, the user programs are made available through the link list.

```
//jobname     JOB job_stmt_parms
//* JOB TO RUN FTP IN BATCH
//EXEC PGM=IKJEFT01,REGION=3000K
//SYSTSPRT   DD SYSOUT=X
//SYSPRINT   DD SYSOUT=X
//SYSPUT     DD *,DCB=BLKSIZE=80
open unix
user pass
mkdir /u/lpn/d.new
//SYSTSIN DD *
FTP2 / option1 option2...
```

Client FTP2 Invocation Options

This section describes the Client FTP2 invocation options and their requirements. These general notes apply to the Client FTP2 invocation options:

- No invocation options are required for the Client FTP2 program. If you specify any, they must be preceded by a slash (/) to meet the conventions of the PL/I runtime support package.
- An option name can immediately follow a slash, as in /FIOS.
- Invocation options are not case sensitive.
- Specify option names exactly as shown. Abbreviations are not permitted.

Client FTP2 General Options

This section describes the Client FTP2 general invocation options.

APP

The APP option (in the form APP**=***vtam_application_name*) identifies the VTAM application name where the client connects. *vtam_application_name* is a 1- to 8-character name.

If the APP option is used, the SYS invocation option is ignored.

FIOS

Normally the FTP program interacts with the user through a terminal. The FIOS option lets the program read and execute a file containing commands and sends the results to a different file. Commands are read from a sequential file allocated to the SYSGET DD statement; execution results are written to a sequential file allocated to the SYSPUT DD statement. Under TSO, the files can be allocated as shown here:

ALLOCATE FILE(SYSGET) DATASET(*input_dataset*)

ALLOCATE FILE(SYSPUT) DATASET(*output_dataset*)

The input file should

- include all the necessary commands (such as **open** and **log**)
- be unnumbered

The output file should have a blocksize of 133.

HOST

The HOST option identifies the remote host with which you want to connect. If you include the remote host name specified here in the NETRC file (userid.FTP.NETRC), the program takes the user ID and password from the NETRC file. Otherwise, you are prompted for a user ID and password.

HOST=*hostname*

Syntax Description

hostname	A variable that specifies the name of the host.

LOGT

The LOGT option displays the current time before each line is sent to the terminal. This option is automatically set if either the test or FIOS option is specified.

NETRC

The NETRC option indicates a NETRC file is to be used to resolve the remote host ID and password. For batch TMP and straight batch FTP2, the netrc parameter must be coded if a NETRC file is to be used. Additionally, straight batch FTP2 requires a //NETRC DD statement in the job stream pointing to the NETRC file. Interactive and batch TMP users can also pre-allocate the NETRC file by coding an ALLOC statement with NETRC as the DD name. This is useful if the default naming conventions for the NETRC file are not being used.

NOA

No Automatic Logon (noa) specifies that there is no automatic connection to the local host (TCPaccess on MVS).

NOFIRE

The NOFIRE (No Firewall) option indicates that client FTP2 should disregard the Firewall-Friendly FTP RFC1579. The default is to implement FRC1579.

Additionally, the **ftp2 firewall** command may be used to turn this support on or off.

SYS

The SYS invocation option, in the form SYS=x, where x is an arbitrary character, identifies an alternate TCPaccess VTAM application to handle the Telnet connections established by the Client FTP2 program. This option is useful in cases where multiple copies of TCPaccess are running concurrently. You can specify SYS=x to access a network server called ACCESx instead of the usual ACCES.

WAIT

The WAIT option causes the terminal keyboard to remain locked while a data transfer is in progress. Normally the keyboard remains unlocked allowing additional commands to be entered during a data transfer.

This option is automatically set if the FIOS option is specified.

Client FTP2 Debugging Options

The Client FTP2 debugging options are used to gather debugging information on the internal operation of the Client FTP2 program and the interactions between the Client FTP2 program and the Server FTPs. Normally the debugging information displays on the terminal, but if the FIOS option is in effect, the information is written to the data set represented by the SYSPUT DD statement.

Note: Use these options only under the direction of TCPaccess software support personnel.

TEST

Use the TEST option to display all requests sent and all responses received on the control connections to each Server FTP. This option can be turned on or off while the program is running by issuing the **debug** command.

Information logged by the TEST option has this format:

TEST - *test debugging information text*

TESTI

Use the TESTI option to obtain local terminal input and output information, as well as to provide the TEST and DISP information.

Information logged by the TESTI option has this format:

TESTI - *testi debugging information text*

VLT

The VLT option turns on tracing of the virtual line terminal sessions associated with the FTP2 session. This option is useful for debugging VTAM problems between the FTP2 session and the TCPaccess address space. The VLT option generates an enormous amount of output. When used interactively, this output comes to the terminal. When used in batch, the output is written to the SYSVLT DD. When you run FTP2 in batch, you must add this DD statement to the JCL:

```
//sysvlt dd   sysout=*,dcb=blksize=133
```

General Client FTP2 Operation

These steps outline the general procedure for using the Client FTP2 program:

- Issue the FTP2 command (with any optional parameters) to log on to the remote host. You are automatically logged on to the local MVS TCPaccess host.
- If you do not specify a HOST**=** option in your FTP2 command line, or if the remote host you specified is not in the NETRC file, you must issue an **open** command for the remote Server FTP.
- If you specify a host name in your **open** command and the host name is in the NETRC file, the user ID and password are taken from the NETRC file. If the host name is not in the NETRC file, you will be prompted for a user ID and password.

  If your user ID and password fail the logon because one or both were entered incorrectly, you must enter the **log** command to sign on to the remote host.
- Set the appropriate file transfer parameters (such as, MODE, STRUCT, or TYPE).
- Perform the desired transfer operation (such as, GET and PUT).

Path Name

A path name is a string that identifies a file to a file system. A path name must contain a device and/or directory name and a file name. The FTP specification does not specify a standard path name convention. Each user must follow the file naming conventions of the file systems involved in the transfer. Consult the System Administrators at the host sites involved in the transfer for file naming conventions.

Many of the Client FTP2 commands take one or more path name arguments. For information about the syntax for MVS path names supported by the TCPaccess Server FTP, read “Data Set Names” on page 5-28 in “Server FTP”.

Client FTP2 Command Conventions

These general notes apply to the Client FTP2 commands:

Program Prompt

To indicate successful completion of most commands, the Client FTP2 program gives a new prompt. However, when a data transfer command is issued, a prompt appears when the operation begins successfully. You can then enter other commands (such as, status requests with the **stat** command) while the operation proceeds.

Completion of Data Transfer

Final completion of the data transfer command is indicated with a message.

Step 1. Testing the Control Connections

You can use the TEST invocation option to see the specific FTP commands and responses sent and received over the control connections as a result of Client FTP2 commands. If you want more information about this test, refer to *RFC 959.*

Case Sensitivity

The Client FTP2 commands are not case sensitive.

Abbreviations in Commands

Abbreviations are permitted if they are not ambiguous. For example, you can type AB for ABORT but you cannot type only A because several commands begin with this letter.

Brackets in Commands

In the examples, parameters enclosed by brackets are optional for the command line. In many cases, if the optional parameters are omitted from the command line, you are prompted for them.

Syntax Conventions

Command words are shown in uppercase and parameters are shown in lowercase. When the actual values for a parameter are given (such as, **debug[on/off])**, they are shown in uppercase.

Example Conventions

In all examples of Client FTP2 input and output in this manual, user entries are shown in boldface type.

The Client FTP2 examples in this manual assume that you have issued **open** and implied **log** commands similar to this example to connect an IBM MVS TSO user to another system:

```
OPEN unix
220 unix FTP server (SunOS 4.0) ready.
Enter name (unix:jim): jim
331 Enter PASS command
Password:
```

The NETRC File

The Client FTP2 program uses information in the NETRC file for automatic login to the remote host. The NETRC file is assumed to be named dsnpref.FTP.NETRC. The dsnpref is the same as the TSO profile prefix at the time of execution of the **ftp2** command. If noprefix is set in the TSO profile, then the dsnpref is the same as the executing TSO user ID. To use a different naming convention for this file, you must pre-allocate it with an NETRC DD statement.

You can specify any number of MACHINE/LOGIN/PASSWORD/ACCOUNT sets to define remote hosts. If you define a machine with no login, password, and/or account, you are prompted for this information when needed. When a user connects to a remote host, if a remote MACHINE/LOGIN/PASSWORD/ACCOUNT set exists for the host, this set logs the user on to the remote host.

The NETRC file also contains macros executed by the FTP2 program. You can define up to twenty macros in the NETRC file. Begin each macro with a MACDEF statement, enter valid Client FTP2 commands, and terminate the macro with an ENDMAC statement. If you define a macro named INIT, it is executed automatically before the Client FTP2 program issues the first prompt. (You do not need to issue a **$** command to execute the INIT macro.) For more information about the MACDEF statement, read "MACDEF" on page 2-45.

Note: The OPEN statement in your SYSGET SYSIN must exactly match the corresponding machine statement in your NETRC. That is, host, port, and route must match.

The INIT MACRO is executed prior to login to the remote host.

If you have not created a NETRC file, you are prompted to supply a user ID and password for the remote host whenever you open a connection to a remote host.

This is the format of the NETRC file; edit this file to specify user IDs, passwords, and accounts or to create a macro.

```
MACHINE remote_host_name
LOGIN userid (for preceding remote host)
PASSWORD password (for preceding remote host)
ACCOUNT account (for the preceding remote host)
MACDEF macro-name
stat
ENDMAC (terminates a macro)
```

If an installation does not require a password and/or an account, these can be omitted. Before using the following Client FTP2 commands, you must log on to a remote host with an **open** command.

CAUTION ! You should use your local access control facility to protect NETRC files since these files can contain valid user ID password combinations for remote hosts. Only the TSO user ID using the NETRC file should be able to read their own NETRC file.

The NETRC file can also be used in batch jobs by using the NETRC invocation along with a NETRC DD pointing to your NETRC data set.

Note: The batch invocation for NETRC does not assume a default FTP.NETRC data set name.

Client FTP2 High-Level Qualifier

The DEFPRFX parameter on the FTP statement in the file APPCFGxx is a server configuration parameter. It tells Server FTP what is the default high level qualifier to be used when starting an FTP server session. You can change the high level qualifier with the appropriate directory commands (**cwd** and **cdup**).

The default high level qualifier used by Client FTP2 after automatic sign-on to the local MVS host usually matches what was specified on the DSNPRFX parameter.

Client FTP2 has a feature that extracts a TSO profile prefix when running under TSO (either interactively or in batch). If a user ID and TSO profile prefix (not set to NOPREFIX) are different, Client FTP2 issues a **CWD** command to change the default directory to the user's TSO profile prefix after the user connects and logs on to the local MVS host. This **cwd** command is not issued if the TSO profile prefix is the same as the TSO user ID or if the TSO profile is set to NOPREFIX.

User ID USER01 has a TSO profile prefix of XXYY. After automatically connecting to and signing on to the local MVS host, this server command is issued by FTP2 to the local MVS host:

CWD 'xxyy.'

Client FTP2 issuing the **cwd** command (when the TSO profile prefix is different from the user ID) may seem to conflict when a user sets DSNPRFX(NONE) on the FTP statement in member APPCFGxx. However, some users use this feature when running Client FTP2. Use this command for a simple work-around to remove the directory prefix when running under Client FTP2:

LQUOTE CDUP

Client FTP2 Commands

"Table 2-1" on page 2-15 gives a brief description of each command and its function; detailed descriptions of each command follow the table.

Table 2-1 Client FTP2 Commands

Command	Description
? [*command_name*]	Get help on all commands or one command
!*tso_command parameters*	Execute TSO command (same as **DO**)
$ *macro_name*	Execute a predefined list of commands
ABOR	Terminate data transfer immediately
ALLO *num_bytes* [**R** *size*]	**ALLO** provides a file size to those Server FTPs that require it. The TCPaccess Server FTP does not require use of the **ALLO** command; use it optionally to truncate records
APPEND [*local_path*] [*remote_path*]	Use **APPEND** to append records to a specified file on the remote host
BINARY	**BINARY** sets the file transfer mode to binary
BYE	Terminate program; same as **END** and **QUIT**
CD [*path*]	Change to the directory specified
CDUP	Change to parent of current working directory
CLOSE	Disconnect from remote host; same as **DISCONNECT**
DEBUG [**ON**|**OFF**]	Display all network commands and responses
DELETE [*path_name*]	Delete file on remote host
DIR [*remote_path*] [*local_path*]	Display contents of the directory specified on the remote host
DISCONNECT	Same as **CLOSE**
DO *tso_command parameters*	Run TSO command; same as !
EBCDIC	Set file transfer type to EBCDIC
END	Terminate program; same as **BYE** and **QUIT**
EXPE	Toggle experimental mode for directory commands
FIREWALL	Toggle implementation of Firewall-Friendly FTP
GET [*remote_path*] [*local_path*]	Copy file from remote host; same as **RECV**
HELP [*text*]	Display local host command information
LOG [*userid*] [*password*] [/*new_password*]	Log in user
LQUOTE [*text*]	Send an FTP command to the local server.
LS [*remote_path*] [*local_path*]	Display names of files in current working directory on remote host
MACDEF *macro_name*	Create a macro

Command	Description
MKDIR [*path_name*]	Create a new directory
MODE S|B	Set transmission mode
NTRANS [*inchars*] [*outchars*]	Translate file names
OPEN [*host_name*]	Open connection to remote host
PUT [*local_path*] [*remote_path*]	Copy file from local host to remote host; same as **SEND**
PWD	Show name of working directory on remote host
QUIT	Terminate program; same as **BYE** and **END**
QUOTE [*text*]	Send FTP command to remote server
RECV [*remote_path*] [*local_path*]	Copy file from remote host to local host; same as **GET**
REMHELP [*text*]	Ask remote host for command information
REMSITE *text*	Send host-specific information to remote host
REMSNDS	Resend last **REMSITE** command to remote host
REMSTAT [*path_name*]	Display status of remote host
RENAME [*old_path_name*] [*new_path_name*]	Rename file on remote host
REST *local_marker remote_marker*	Restart
RMDIR [*path*]	Remove directory on remote host
SEND [*local_path*] [*remote_path*]	Copy file from local host to remote host; same as **PUT**
SITE *text*	Send information specific to local host
SNDS	Resend last **SITE** command to local host
STATUS [*parameter*]	Display status of local host
STRUCT F|R	Set file structure
SUNIQUE	Store unique file names on remote host
TYPE I | L *byte_size* | { **A** | **E** [**N** |**T** | **C**] }	Set data type

? Command

Use the **?** command to display information about using the Client FTP2 program.

? [*command_name*]

Syntax Description

command_name	Command for which you are requesting information.

Default

If the **?** command is specified with no arguments, it displays a list of Client FTP2 commands.

Usage Guidelines

- The **?** command can be used either with or without arguments.
- If *command_name* is specified as an argument to the **?** command, a line of information displays similar to the information in the table in "Client FTP2 Commands" on page 2-15. The line shows the command syntax and a short description of the command function.

Example

```
? Dir
```

DIR <*pathr*> <*pathl*> - Directory list of remote host

Related Commands

"HELP" on page 2-40	Requests help from your local Server FTP.
"REMHELP" on page 2-56	Requests help from the remote Server FTP.

! Command

The **!** command (do **tso** command) requests the Client FTP2 program to execute a **tso** command for you.

! *TSO_command parameters*

Syntax Description

tso_command parameters	Lists the parameters to be passed to the **tso** command.

Usage Guidelines

- The *tso_command* argument is required.
- This command requires that the TSO environment exist.
- The **!** command cannot be used for batch FTP2 without the TMP.

Example

```
! listc
IN CATALOG: CATALOG.MVSICF1.VMVSTSO
USER1.ACCES.ASM
USER1.LIB.LOAD
USER1.T.D
USER1.VBIG.D
USER1.VB.D.
```

$ Command

The **$** command executes a specified macro you have defined either with a **macdef** command or in the NETRC file (see the sections of this chapter describing the **macdef** command and the NETRC file for more information).

$ *macro_name*

Example

In the following example, **$** is calling a macro called **sp**, which displays status, the current working directory, and ends the macro.

```
macdef sp
SP:stat
SP:pwd
SP:endmac
$ sp
EXECUTING: stat
Connected to: UNIX
--- STATUS ---
-- FTP Parameters ---
Remote DT Host, Port 127.0.0.1, 0
Local DT  Host, Port 138.42.224.15, 0
Type A N   Tabs 8   Stru F   Mode S   Recall 5
-- END --
-- Control --
User USER1   Acct Accs E0000200   Unit SYSALLDA
Host 138.42.224.15
-- End Control --
-- Path Data --
Rlse
-- End Path Data --
-- Transfer Information --
Data transfer not in progress
-- END --
211 <End of Status>
Executing: pwd
257 "/home/unix/user1" is current directory.
Executing: endmac
```

ABOR

Use **abor** (abort) to instruct the Server FTP to abort the last command issued and any associated transfer of data. No action is taken by the Server FTPs if the previous command has been completed (including data transfer). The control connections to the Server FTPs are not closed, but the data connections are closed.

ABOR

Syntax Description

This command has no arguments or keywords.

Default

The **abor** request is directed to both Server FTPs.

Example

```
get +outmail temp.data
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Path USER1.TEMP.DATA
Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 3120   Space 1 5 Tracks Rlse
150 ASCII data connection for +outmail (138.42.224.15,4099) (167076 bytes)
ABOR
-Data transfer aborted.
 65536 bytes received in 6.65 seconds (9855 bytes/s)
Path USER1.TEMP.DATA   User USER1   Data bytes received 68948
Disk tracks written 1   Records padded 2997
226 Abort command completed.
426 Transfer aborted. Data connection closed.
226 Abort successful
225 ABOR command successful.
```

ALLO

The **allo** command allocates a file size to those Server FTPs that require it. Refer to the documentation for your particular Server FTP to see if you need this command. The TCPaccess Server FTP does not require use of the **allo** command; you can use it to cause records to be truncated or to allocate space.

ALLO *num_bytes* [**r** *size*]

Syntax Description

num_bytes	Number of bytes (using the logical byte size) of storage to be reserved for the file.
r *size*	Maximum size of storage required.

Usage Guidelines

- If files are sent with record or page structure, a maximum record or page size (in logical bytes) can be required.
- The R *size* argument is optional; if specified it must be separated from the first argument by the three characters, " R " (space R space)
- The **allo** command requires a host prefix. For information on host prefixes, see "Using Host Name Strings" on page 1-5.

Example

```
stru r
site lrecl(60) blk(6000)
allo 12000 r 60
put n.d allor.data
ALLOCMD = ALLO 12000 r 60
150-Data set open with attributes:
Type A N   Tabs 8   Stru R   Mode S   Recall Path USER1.ALLOR.DATA
Volser ICSPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 6160   Space 1 1 Tracks Rlse   Maxr 60
-Data set open with attributes:
Type A N   Tabs 8   Stru R   Mode S   PATH USER1.N.D
Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 3120   Rlse
226-Transfer complete
 3441 bytes received in 0.37 seconds (9300 bytes/s)
Path USER1.ALLOR.DATA   User USER1   Data bytes received 3277
Disk tracks written 1   Records truncated 21   Records padded 80
-Transfer complete
 3441 bytes sent in 0.25 seconds (13764 bytes/s)   Path USER1.N.D
User COLMBIA   Data bytes sent 6480
Disk tracks read 1
```

APPEND

The **append** command requests that a file from the local host be appended to a file at the remote host.

APPEND [*local_path*] [*remote_path*]

Syntax Description

local_path	Name of the file to be retrieved from the local host.
remote_path	Remote file to which the local file is to be appended.

Default

If either file name is omitted, you are prompted for the file name.

Usage Guidelines

The syntax for each path depends on the associated Server FTP.

Example

```
append n.d myappend.data
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Path USER1.N.D
Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 3120   Rlse
150-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5
Path USER1.MYAPPEND.DATA
Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 6160   Space 5 3 Tracks Rlse
-Transfer complete
 3439 bytes sent in 0.46 seconds (7476 bytes/s)   Path USER1.N.D
User COLMBIA   Data bytes sent 6480
Disk tracks read 1
226-Transfer complete
 3439 bytes received in 0.42 seconds (8188 bytes/s)
Path USER1.MYAPPEND.DATA   User USER1   Data bytes received 3277
Disk tracks written 1   Records padded 80
```

ASCII

The **ascii** command sets the data type to ASCII for the transfer of an ASCII file.

ASCII

Example

```
ASCII
stat
Connected to: MVS
--- STATUS ---
--- FTP PARAMETERS ---
Remote DT Host, Port 138.42.224.13, 0
Local DT  Host, Port 138.42.224.15, 0
Type A N   Tabs 8   Stru F   Mode S   Recall 5  Server is passive
-- END --
-- Control --
User COLMBIA   Acct Accs E0000200   Unit SYSALLDA   Host 138.42.224.15
-- End Control --
-- Path Data --
Rlse
-- End Path Data --
-- Transfer Information --
Data transfer not in progress   Data bytes sent 6480
Disk tracks read 1   Network bytes sent 3439   Elapsed time 00.00.00
Bytes/Second 7476
-- END --
211 <End of Status>
```

Related Commands

“TYPE” on page 2-70	This command is equivalent to a **type** command with the ASCII (**A**) parameter specified.

BINARY

The **BINARY** command sets the data type to binary for the transfer of a binary file.

BINARY

Syntax Description

This command has no arguments or keywords.

Example

```
BINARY
stat
Connected to: MVS
--- STATUS ---
-- FTP Parameters --
Remote DT Host, Port 138.42.128.13, 0
Local DT  Host, Port 138.42.224.15, 0
Type I N   Stru F   Mode S   Recall 5   Server is passive
-- END --
-- Control --
User COLMBIA   Acct Accs E0000200   Unit SYSALLDA   Host 138.42.224.15
-- End Control --
-- Path Data --
Rlse
-- End Path Data --
-- Transfer Information --
Data transfer not in progress   Data bytes sent 6480
Disk tracks read 1   Network bytes sent 3439   Elapsed time 00.00.00
Bytes/Second 7476
-- END --
211 <End of Status>
```

Related Commands

"TYPE" on page 2-70	This command is equivalent to a **type** command with the image (**I**) parameter specified.

BYE

The **BYE** command terminates the Client FTP2 program. This command is the same as the **END** and **QUIT** commands.

BYE

Syntax Description

This command has no arguments or keywords.

Example

BYE

```
221 Goodbye.
```

Related Commands

"END" on page 2-35	Can be used instead of **BYE**; **END** does not require a host prefix.
"QUIT" on page 2-53	Can be used instead of **BYE**; **QUIT** requires a host prefix and takes no arguments.

CD

The **CD** command requests that the remote Server FTP change the current directory to a new directory.

CD [*path_name*]

Syntax Description

path_name	Indicates to the remote Server FTP the name of the directory to be made the current directory.

Default

If you omit *path_name*, the Client FTP2 program prompts you for the desired value.

Usage Guidelines

The syntax for *path_name* depends on the associated Server FTP.

Example

A UNIX Server FTP in session with the Client FTP2 program is using /u/user1/work as the current directory. If a **CD junk** command is issued by the Client FTP2 to that UNIX Server FTP, the resulting current directory is /u/user1/work/junk. The same result is achieved by specifying **CD** /u/user1/work/junk.

This example shows a change of directory to a remote UNIX system:

CD /u/lpn/d.ddn

```
250 CWD command successful.
```

This example shows a change to a directory on a remote system running TCPaccess:

CD acces

```
250 "'USER1.ACCES.'" is current prefix
```

CDUP

The **CDUP** command directs the remote Server FTP to change the current directory to the parent directory of the old current directory. The **CDUP** command is most useful when the Server FTP manipulates a hierarchical file system such as UNIX.

CDUP

Syntax Description

This command has no arguments or keywords.

Example

A UNIX Server FTP in session with the Client FTP2 program has */u/user1/work* as the current directory. If a **CDUP** command is issued by the Client FTP2 to that UNIX Server FTP, the resulting current directory is the parent of the old current directory (*/u/user1*).

This example shows a change to the parent directory of the old current directory on a remote UNIX system:

```
CDUP
250 CWD command successful.
```

CLOSE

The **CLOSE** command logs you out and terminates the connection between you and the remote Server FTP.

CLOSE

Syntax Description

This command has no arguments or keywords.

Example

```
CLOSE
221 Goodbye.
```

DEBUG

The **DEBUG** command displays all commands and responses going to and from the Server FTPs and the user. This command is equivalent to specifying the TEST option on the Client FTP2 command line.

DEBUG [ON | OFF]

Syntax Description

DEBUG ON	Enables you to see all commands and responses going to and from the Server FTPs.
DEBUG OFF	Turns off the DEBUG option.

Default

If no parameter is specified with the **DEBUG** command, the command toggles the previous state.

Example

```
DEBUG ON
```

DELETE

The **DELETE** command directs the remote Server FTP to delete the specified file.

DELETE [*path_name*]

Syntax Description

path_name Specifies the specific file to delete.

Default

If you omit *path_name*, you are prompted to supply one.

Usage Guidelines

The syntax for *path_name* depends on the associated Server FTP.

Example

```
DELETE gatedel
250 DELE command successful.
```

DIR

The **DIR** command requests that the remote Server FTP provide a directory list for the specified path.

DIR [*remote_path*] [*local_path*]

Syntax Description

remote_path	specifies the remote directory to be listed. If *remote_path* is not specified, you receive a list of your current directory
local_path	specifies the file to which the directory list is written. If *local_path* is not specified, the directory list is displayed on your screen

Defaults

The current directory is displayed to your screen.

Usage Notes:

The **DIR** command requires that the TSO environment exist (that is, you are not running batch FTP2 without the TMP).

Example

The following example lists the contents of a directory on an MVS system.

```
ftp> DIR /export/home/user1/mvs
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5
Path USER1.FTP.TMP.T1330492
Volser ICSUSR   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 6160   Space 5 3 Tracks Rlse
150 ASCII data connection for /bin/ls (138.42.224.15,4126) (0 bytes).
226 ASCII Transfer complete.
-Transfer complete
 322 bytes received in 1.50 seconds (214 bytes/s)
Path USER1.FTP.TMP.T1330492   User USER1   Data bytes received 310
Disk tracks written 1   Records padded 6
total 20
-rw-------     1 user1     dvlp     2730 Oct 22     08:36 channel
-rw-------     1 user1     dvlp     2901 Sep  1     09:17 comten
-rw-------     1 user1     dvlp     1276 Oct 21     14:43 dump
-rw-------     1 user1     dvlp      729 Nov 12     05:24 ibmlink
-rw-------     1 user1     dvlp      751 Nov 12     07:41 ibmlink2
IDC05501 ENTRY (A) USER1.FTP.TMP.T1330492 DELETED
ftp>
```

DISCONNECT

The **DISCONNECT** command disconnects you from a host. It logs you out and terminates the connection between you and the remote Server FTP. The CLOSE command is the same as the **DISCONNECT** command.

DISCONNECT

Syntax Description

This command has no arguments or keywords.

Example

The following example shows the **DISCONNECT** command.

```
ftp> DISCONNECT

disconnect
221 Goodbye.
ftp.
```

DO

The **DO** (do TSO command) command requests the Client FTP2 program to execute a TSO command for you.

DO *tso_command parameters*

Syntax Description

tso_command parameters	Specifies the TSO command followed by any parameters to be passed to the TSO command.

Usage Notes

- The *tso_command* argument is required.
- The **DO** command is handled by the Client FTP2 program.
- The **DO** command requires that the TSO environment exist (i.e., you are not running batch FTP2 without the TMP).

Example

```
DO listc l(lpn)
NONVSAM-------LPN.ABOR
IN-CAT--CATALOG.MVSICF1.VMVSTSO
NONVSAM-------LPN.FTPNETRC
IN-CAT--CATALOG.MVSICF1.VMVSTSO
NONVSAM-------LPN.CNTL
IN-CAT--CATALOG.MVSICF1.VMVSTSO
NONVSAM-------LPN.FTPEXAMPLE
IN-CAT--CATALOG.MVSICF1.VMVSTSO.
```

EBCDIC

The **EBCDIC** command sets the data type to EBCDIC for transfer of an EBCDIC file. The command is equivalent to the **TYPE** command with the EBCDIC (**E**) parameter specified.

Example

```
EBCDIC
EBCD ENTERED
stat
CONNECTED TO: MVS
--- FTP PARAMETERS ---
REMOTE DT HOST,PORT 129.192.224.136, 0
LOCAL DT HOST,PORT 129.192.224.136, 20
TYPE E N TABS 8 STRU F MODE S
--- ACCESS CONTROL ---
USER LPN ACCT ACCS E0000200 VOLS MVSTSO TELNET HOST 129.192.224.136
SESSION# 412
--- PATH DATA ---
--- TRANSFER INFORMATION ---
DATA TRANSFER NOT IN PROGRESS
```

END

The **END** command terminates the Client FTP2 program. This is typically the last command you enter. Any open control connections are closed before the program terminates.

END

Syntax Description

This command has no arguments or keywords.

Example

```
END
221 Session terminated
```

Related Commands

“BYE” on page 2-25	Can be used instead of **END**. **BYE** works exactly as the **QUIT** command.
“QUIT” on page 2-53	Can be used instead of **END**; **QUIT** requires a host prefix and takes no arguments.

EXPE

The **EXPE** command toggles the use of experimental or regular directory commands. Since there is no consistent support for this command, it is recommended that you not use this command.

EXPE

Defaults

No operands are associated with the **EXPE** command since it is a Client FTP2 command.

Usage Guidelines

"Table 2-2" on page 2-36 shows the FTP command that is sent over the control connection for each directory command with an **EXPE** setting:

Table 2-2 Using Directory Commands with an EXPE Setting

Client FTP2 Command	Regular	Experimental
MKD	MKD	XMKD
RMD	RMD	XRMD
PWD	PWD	XPWD
CDUP	CDUP	XCUP

Note: The directory commands were added to FTP subsequent to the initial FTP specification and are documented in *RFC 959*, File Transfer Protocol (FTP) Appendix II, Directory Commands.

FIREWALL

The **FIREWALL** command toggles the implementation of Firewall-Friendly FTP RFC **1579** on and off. RFC **1579** specifies that for FTP data connection establishment, the client sends the **PASV** command to the remote and the **PORT** command to the local host. This is the default. When the **FIREWALL** command is entered to disable this implementation, the FTP2 client will send the **PASV** command to the local and the **PORT** command to the remote. You can turn the implementation back on by entering **FIREWALL** again. A message is returned to the user indicating if Firewall is enabled or disabled.

FIREWALL

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

No arguments are associated with the **FIREWALL** command.

Note: This option may also be turned off via the NOFIRE parameter of the Client FTP2 Invocation Options

Example

The following example shows the effect of turning the firewall on and off. Initially, the firewall implementation is on.

```
10:06:23   FTP2:  put n.d temp
10:07:26     TEST - SENDING - B:PASV
10:07:27   227 Entering Passive Mode (138,42,32,165,145,225)
10:07:27     TEST - SENDING - A:PORT 138,42,32,165,145,225
10:07:28   200 OK, Ready
10:07:28     TEST - SENDING - A:RETR n.d
10:07:29   150-Dataset open with attributes:
10:07:29   Type A N    Tabs    8   Stru F   Mode S   Path ABC.N.D   Volser ICS007
10:07:29   Unit SYSICS   Dsorg        PS   Recfm FB   Lrecl   80   Blksize   3120
10:07:29   Rlse
10:07:30   150
10:07:30     TEST - SENDING - B:STOR temp
10:07:31   150 ASCII data connection for temp (138.42.220.13,20).
10:07:31   FTP2:
10:07:34   226-Transfer complete
10:07:35    3439 bytes sent in 1.17 seconds (2939 bytes/s)   Path ABC.N.D
10:07:35   User ABC Data bytes read 6480
10:07:36   Disk tracks read 1
10:07:36   226
10:07:36   226 Transfer complete.
10:07:36   FTP2:  firewall
10:10:28   ACC831I - Firewall has been disabled
10:10:28   FTP2:  put n.d temp
10:10:58     TEST - SENDING - A:PASV
10:11:00   227 Entering passive mode    138,42,220,13,20,27
10:11:00     TEST - SENDING - B:PORT 138,42,220,13,20,27
10:11:01   200 PORT command successful.
```

```
10:11:01     TEST - SENDING - A:RETR n.d
10:11:02   150-Dataset open with attributes:
10:11:03   Type A N    Tabs    8   Stru F   Mode S   Path ABC.N.D   Volser ICS007
10:11:04   Unit SYSICS   Dsorg        PS   Recfm FB   Lrecl   80   Blksize   3120
10:11:04   Rlse
10:11:04   150
10:11:04     TEST - SENDING - B:STOR temp
10:11:05   150 ASCII data connection for temp (138.42.220.13,5147).
10:11:05   FTP2:
10:11:27   226-Transfer complete
10:11:28    3439 bytes sent in 1.22 seconds (2818 bytes/s)   Path ABC.N.D
10:11:28   User ABC   Data bytes read 6480
10:11:28   Disk tracks read 1
10:11:28   226
10:11:28   226 Transfer complete.
10:11:28   FTP2:  firewall
10:13:10   ACC831I - Firewall has been enabled
10:13:10   FTP2:
```

GET

The **GET** command requests that a file from the remote host be copied to a file on the local host by the appropriate Server FTPs. The file to be retrieved is always at the remote host, and the file to be copied into is always at the local host.

GET [*remote_path*] [*local_path*]

Syntax Description

remote_path	Specifies the file name to be retrieved from the remote host.
local_path	specifies the file name at the local host into which the file from the remote host is copied.

Defaults

If you omit either file name, you are prompted for one.

Usage Guidelines

The syntax for each path depends on the associated Server FTP.

Note: The **FTP GET DGDBASE** command gets only the most recent data set.

Check with your TCPaccess Administrator to find out if existing data sets are allowed to be overwritten (this is specified on the FTP statement in APPCFG*xx*).

Example

The following example gets the file comten from the remote host and saves it on the local host as temp.data:

```
ftp> GET comten temp.data
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S  Recall 5  Path USER1.TEMP.DATA
Volser HAGCAT   Unit SYSALLDA   Dsorg  PS   Recfm FB   Lrecl  80
Blksize 3120   Space 1 5 Tracks Rlse
150 ASCII data connection for comten (138.42.224.15,4127) (2091 bytes).
-Transfer complete
 2291 bytes received in 1.34 seconds (2232 bytes/s) Path USER1.TEMP.DATA
User USER1   Data bytes received 2825
Disk tracks written 1   Records padded 90   Records folded 2
226 ASCII Transfer complete.
ftp>
```

Related Commands

“RECV” on page 2-55	This command is the same as the **GET** command.

HELP

The **HELP** command requests help information from the local Server FTP.

HELP [*text*]

Syntax Description

The **HELP** command has no required operands. Any operands specified on the **HELP** command are passed through unchanged to the Server FTP and are interpreted by the Server FTP.

Usage Guidelines

- After a **REST** command, **STOR** and **APPE** have identical meanings.
- Data transfer must be MODE B (block).
- A file retrieved normally includes restart markers approximately every 32767 bytes. The **REST** parameter on the **SITE** command allows the user to change this interval or even entirely suppress restart markers. See **HELP SITE**. The actual decision to send a marker depends on a count of data bytes read from the disk (not including OS count/control bytes). When this count reaches the limit, the marker is sent at the next end of a complete logical record, segment of a spanned record (if **RECFM** includes **VS**), or a physical disk block (if RECFM is U, V, or F).
- **FTP** can accept (and send) restart markers in either STRU F or STRU R.
- **FTP** restart markers consist of 10 characters, which are the hexadecimal representation of five 8-bit bytes: *TTRBB*. Here, ***TTR*** forms a standard OS disk block address, and *BB* is a byte offset within the block.

Example

```
HELP REST
--- HELP---
*** HELP REST ***
FTP REST (Restart) Command:
Function: Specifies that the data transfer command that follows  (immediately)
is to restart
at a specified intermediate point in the file.
Syntax:  REST <marker>
214  <end of HELP>
```

Related Commands

"? Command" on page 2-17	Requests help from the Client FTP2 program.
"HELP" on page 2-40	Requests help from your local Server FTP.
"REMHELP" on page 2-56	Requests help from the remote Server FTP.

LOG

The **log** (login) command gives a user ID and password to a remote Server FTP to identify the user. You can optionally change your password when logging in to the TCPaccess server. The **log** command typically is issued immediately following the **open** command.

LOG [*userid*] [*current_password*] [*/new_password*]

Syntax Description

userid	User name.
current_password	Password for the user name.
new_password	New password if you choose to change passwords upon logging in.

Default

The Client FTP2 program prompts you for the user ID and/or current password.

Usage Guidelines

If the remote Server FTP requires additional accounting information during the user identification process, the Client FTP2 program prompts you to enter the accounting data.

Examples

The /new_password parameter is a 1- to 8-character string password. The new password replaces the current password after the user ID and current password are validated. The new password option is valid only when talking to the TCPaccess server. The slash (/) must follow the current password without any intervening blanks. The new password must follow the slash without any intervening blanks, as in the following example:

```
LOG lpn tstpass
230 User lpn logged in.
```

In the following example, user USER01 wants to change his current password from CJAY to MACDUFF.

```
LOG user01 cjay/macduff
230 User USER01 logged in.
```

LQUOTE

The **lquote** command sends an uninterpreted, unaltered character string to the local Server FTP over the control connection. This mechanism sends FTP commands to the Server that the Client FTP2 program might not be able to send.

LQUOTE [*text*]

Syntax Description

This command has no arguments or keywords.

Default

If the text is omitted, you are prompted for it.

Usage Guidelines

The text is sent to the local FTP Server over the control connection exactly as you entered it.

Example

```
LQUOTE site vol(mvsts2)
```

LS

The **ls** command requests a remote Server FTP to provide a list of file names for the specified path.

LS [*remote_path*] [*local_path*]

Syntax Description

remote_path	Path to be listed from the remote host.
local_path	Local file into which the list from the remote server is printed.

Default

If a local path is not specified, the list of files appears on your screen.

Usage Guidelines

- If the path specifies a directory or other group of files, the remote Server FTP transfers a list of files.
- The syntax for each path depends on the associated Server FTP
- If a local path is specified, the list of files is written to the specified file.
- The **LS** command requires that the TSO environment exist (that is, you are not running batch FTP2 without the TMP).

Examples

The following example shows the **ls** command with parameters:

```
LS /export/home/user1 unix.dir.temp
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5
Path USER1.UNIX.DIR.TEMP
Volser ICSUSR   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl  80
Blksize 6160   Space 5 3 Tracks Rlse
150 ASCII data connection for /bin/ls (138.42.224.15,4134) (0 bytes).
226 ASCII Transfer complete.
-Transfer Complete
 350 bytes received in 1.54 seconds (227 bytes/s)
Path USER1.UNIX.DIR.TEMP   User USER1   Data bytes received 274
Disk tracks written 1   Records padded 38
```

The following example shows the **LS** command without parameters:

```
LS
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5
Path USER1.FTP.TMP.T1443273
Volser ICSUSR   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl  80
Blksize 6160   Space 5 3 Tracks Rlse
150 ASCII data connection for /bin/ls (138.42.224.15,4135) (0 bytes).
226 ASCII Transfer complete.
-Transfer complete
 42 bytes received in 1.47 seconds (28 bytes/s)
Path USER1.FTP.TMP.T1443273   User USER1   Data bytes received 32
Disk tracks written 1   Records padded 5
```

```
channel
comten
dump
filea
tempfile
IDC0550I ENTRY (A) USER1.FTP.TMP.T1443273 DELETED
```

MACDEF

The **MACDEF** command enables you to define a sequence of commands you want to execute more than once (a macro).

MACDEF *macro_name*

Syntax Description

macro_name	Name of the macro being defined.

Usage Guidelines

- When you issue the **MACDEF** command, you are prompted for each line of input. You must enter valid Client FTP2 commands.
- To terminate input, press **RETURN**.
- To execute the macro, issue a **$** command (read "$ Command" on page 2-19).
- The macro is temporary and is deleted automatically at the end of your session. To create a permanent macro, read "The NETRC File" on page 2-13.
- **MACDEF** handles \ (backslash) and $ (dollar) as special characters.
- A $ followed by a digit 1-9 is replaced by the corresponding argument on the macro invocation command line (the $ command).
- A \ followed by any character is replaced by that character. Use \ with $ to prevent special handling of the $.
- Use \\ to cause a single \ to be interpreted.
- Macros cannot be nested (one macro cannot call another macro).

Note: A **NETRC** file must be used with the following example for login to the remote host.

Example

```
USERFTP: MACDEF getfrom
GET:open $1
GET:pwd
GET:get $2 $3
GET:close
GET:endmac
USERFTP: $ getfrom host1.md.company.com ftpdata 'abc.ftp.test'
Executing: open host.md.company.com
220 host FTP server (UNIX(r) System V Release 4.0) ready.
230 User abc logged in.
Executing: pwd
257 "/opt/home/abc" is current directory.
Executing: get ftpdata 'abc.ftp.test'
-Dataset open with attributes:
Type A N  Tabs  8   Stru F   Mode S   Recall 5   Path ABC.FTP.TEST
Volser ICSPK3  Unit SYSALLDA  Dsorg  PS   Recfm U
Blksize   6160    Space 1     Tracks   Rlse
150 ASCII data connection for ftpdata (138.42.128.13,4679)
   (2893 bytes
226 ASCII Transfer complete.
-Transfer complete
2927 bytes received in 1.16 seconds (2523 bytes/s)
  Path  ABC.FTP.TES
User ABC  Data bytes written 2927
Disk tracks written 1
Executing: close
Executing: endmac
```

MKDIR

The **MKDIR** (make directory) command directs a remote Server FTP to create the specified directory.

MKDIR [*path_name*]

Syntax Description

path_name	Directory to be created.

Defaults

If you omit *path_name*, you are prompted for it.

Usage Guidelines

- If the path name is relative, the specified subdirectory is created in the current working directory.
- If the path name is absolute, the specified directory is created.
- The syntax for path depends on the associated Server FTP.

Example

A UNIX Server FTP in session with the Client FTP2 program has */u/user1/work* as the current directory. If a **MKDIR junk** command is issued by the Client FTP2 to that UNIX Server FTP, the subdirectory junk is created in the current directory (*/u/user1/work/junk*). The same result is achieved by specifying **MKDIR** */u/user1/work/junk*.

```
MKDIR /u/lpn/d.new
257 MKD command successful.
```

MODE

The **mode** command sets one of two transmission modes:

- Block mode

Block mode formats the data and allows for restart procedures.

- Stream mode

Stream mode passes the data with little or no processing. It interacts with the structure attribute to determine the type of processing. Stream mode is the default if no **MODE** command was used.

For the purpose of standardized transfer, the sending host translates its internal end-of-line or end-of-record representation into the representation required by the transfer mode and file structure, and the receiving host performs the inverse translation to its internal representation. Since these transformations make extra work for some systems, identical systems transferring non-record structured text files might use binary representation and stream mode to simplify transfer.

MODE S | B

One of the two codes (either s or b) is required as an argument on the **mode** command.

Each of the possible transmission modes is discussed in the following sections. For a detailed description of the effect of various transmission modes, read the Transmission Modes section in *RFC 959*, File Transfer Protocol.

Not all Server FTPs support all transmission modes; review the Server FTP documentation if you have questions concerning transmission mode support.

Block Mode

Block mode is indicated with the character **b**. In block mode, the file is transmitted as a series of data blocks preceded by one or more header bytes. Record structures are allowed in this mode, and any representation type can be used. Restart markers are embedded in the data stream.

Stream Mode

Stream mode is set with the character **s**. This is the default if no **mode** command has been used. In stream mode, the data is transmitted as a stream of bytes. There are no restrictions on the representation type used, and record structures are allowed. In a record structured file, End of Record (EOR) and End of File (EOF) are each indicated by a two-byte control code included with the data sent over the data connection. If the structure is a file structure, the EOF is indicated by the sending host closing the data connection, and all bytes sent over the data connection are data bytes.

NTRANS

The **ntrans** command is executed prior to a file transfer to enable you to change the name of the file you are transferring. When you execute the file transfer and create a file at a destination, the transferred file has the name you specify with the **ntrans** command.

NTRANS [*inchars*] [*outchars*]

Syntax Description

inchars	Specifies the characters in the file name that you want to change.
outchars	Specifies the new characters.

Default

If either parameter is omitted, you are prompted for it.

Example

Suppose that you have a file named ABC(DEF) and you issue this command:

```
NTRANS () .z
```

To effect the change specified in the **ntrans** command, issue a **put** or **get** command to transfer the file, as shown here:

```
PUT abc(def)
```

Issue the file transfer command without a second file name. The file ABC(DEF) is created at its destination with the name ABC.DEFZ (the left parenthesis is changed to a period (.) and the right parenthesis is changed to a Z).

```
NTRANS nd ab
PUT n.d
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Path USER1.N.D
Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 3120 Rlse
150-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5   Path USER1.A.B
Volser ICSPK2   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 6160   Space 5 3 Tracks Rlse
-Transfer complete
 3439 bytes sent in 0.55 seconds (6252 bytes/s)   Path USER1.N.D
User USER1   Data bytes sent 6480
Disk tracks read 1
226-Transfer complete
 3439 bytes received in 0.48 seconds (7164 bytes/s) Path USER1.A.B
User USER1   Data bytes received 3277
Disk tracks written 1   Records padded 80
```

OPEN

The **open** command sets up one control connection between the Client FTP2 program and a Server FTP. You can also connect to a “remote” Server FTP that is actually your local Server FTP by specifying your own local host name.

OPEN [*host_name*]

Syntax Description

host_name	Name of the remote host.

Default

If you omit the host name, you are prompted for one.

Usage Guidelines

- Host name strings must correspond to the syntax specified in “Introduction to TCPaccess”.
- Once a host connection is established, FTP2 prompts for a user ID, password (and optionally an account) combination when one is not provided in the NETRC file.
- A successful **open** command implies an automatic **log** command attempt by FTP2.

Example

```
OPEN unix
220 unix FTP server (SunOS 4.0) ready.
Enter name (unix:user1): user1
331 Enter PASS command
Password:
```

PUT

The **put** command requests that the appropriate local Server FTP copy a file to the remote system. The file to be copied is always at the local Server and the file destination is always at the remote Server.

PUT [*local_path*] [*remote_path*]

Syntax Description

local_path	File name of the file to be copied from the local server.
remote_path	File name of the file at the remote Server into which the file from the local server is copied.

Default

If you omit either path, you are prompted for the file name.

Usage Guidelines

- The command name **SEND** can be used in place of **PUT**. There are no other differences when using **SEND**.
- The syntax for each path depends on the associated Server FTP.

Example

```
PUT telnet.data teltemp
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Path USER1.TELNET.DATA
Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 120
Blksize 3600   Rlse
150 ASCII data connection for teltemp (138.42.224.15,4104)
-Transfer complete
 11535 bytes sent in 2.22 seconds (5195 bytes/s)
Path USER1.TELNET.DATA
User COLMBIA   Data bytes sent 19680
Disk tracks read 1
226 ASCII Transfer complete.
```

PWD

The **pwd** command directs a remote Server FTP to return the path name of the current working directory.

PWD

Syntax Description

This command has no arguments or keywords.

Example

```
PWD
257 "/u/lpn" is current directory.
```

QUIT

The **quit** command terminates the Client FTP2 program. This command is the same as the **bye** and **end** commands.

QUIT

Syntax Description

This command has no arguments or keywords.

Example

```
quit
221 Goodbye.
```

Related Commands

“BYE” on page 2-25	Can be used instead of **QUIT**. **BYE** works exactly as the **QUIT** command.
“END” on page 2-35	Can be used instead of **QUIT**; **END** does not require a host prefix.

QUOTE

The **quote** command sends an uninterpreted, unaltered character string to the remote Server FTP over the control connection. This mechanism sends FTP commands to the Server that the Client FTP2 program might not be able to send.

QUOTE [*text*]

Syntax Description

text	Text sent to the Server over the control connection exactly as you enter it.

Default

If the text is omitted, you are prompted to enter it.

Example

```
QUOTE pasv
227 Entering Passive Mode (26,131,0,17,4,216).
```

RECV

The **recv** command requests that a file from the remote host be copied to a file on the local host by the appropriate Server FTPs. The file to be retrieved is always at the remote host, and the file to be copied to is always at the local host.

RECV [*remote_path*] [*local_path*]

Syntax Description

remote_path	File name to be retrieved from the remote host.
local_path	File name at the local host that the file from the remote host is copied into.

Default

If you omit either file name, you are prompted for it.

Usage Guidelines

The syntax for each path depends on the associated Server FTP.

Note: Check with your TCPaccess Administrator to find out if existing data sets are allowed to be overwritten (this is specified on the FTP statement in APPCFG*xx*).

Example

```
RECV teldata temp.data
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5   Path USER1.TEMP.DATA
Volser ICSUSR   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 6160   Space 5 3 Tracks Rlse
150 ASCII data connection for teldata (138.42.224.15,4112) (11371 bytes).
226 ASCII Transfer complete.
-Transfer Complete
 11535 bytes received in 1.56 seconds (7394 bytes/s)
Path USER1.TEMP.DATA   User USER1   Data bytes received 11207
Disk tracks written 1   Records padded 147   Records folded 69
```

Related Commands

"GET" on page 2-39 The **recv** command is the same as the **get** command.

REMHELP

The **remhelp** command requests help information from the remote Server FTP

REMHELP [*text*]

Syntax Description

text	Any text specified on the **remhelp** command is passed through unchanged to the remote Server FTP and is interpreted by the remote Server FTP.

Usage Guidelines

- No operands are required for the **remhelp** command.
- The **remhelp** command is different from the **?** command in that the **remhelp** command requests help from the remote Server FTP, whereas the **?** command requests help from the Client FTP2 program.

Example

```
REMHELP list
214 Syntax: LIST .  <sp> path-name .
```

REMSITE

The **remsite** (remote site parameters) command provides the remote Server FTP with specific information it requires. This information is essential to file transfers involving that Server FTP, but is not sufficiently universal to have been included specifically in the FTP. Typically, you use a **remhelp site** Client FTP2 command to find the site requirements for a specific remote Server FTP. Otherwise, review the Server FTP documentation for the site requirements.

REMSITE *text*

Syntax Description

text	Text that is required and is passed through unchanged to the specified server.

Example

The following example shows a **remsite** command to change the default FTP volume:

```
REMSITE vol(mvsts2)
```

Note: The *vol* parameter is specific to an MVS remote server.

REMSNDS

The **remsnds** (resend **REMSITE** parameters) command directs the Client FTP2 program to resend the last **remsite** command to the remote Server FTP. Your **remsite** command is reissued without your having to retype it. Since most Server FTPs require that new site parameters be provided before each data transfer, the **remsnds** saves time if you repeatedly use identical site parameters.

REMSNDS

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

You can preserve **site** parameters between data transfers by using the PERSIST option in the **site** command.

Example

```
SITE vol(mvsts2)
REMSNDS
site vol(mvsts2) <Sent
```

REMSTAT

The **REMSTAT** command requests a status response from the remote system.

REMSTAT [*path_name*]

Syntax Description

path_name	Path on the remote host from which status is requested.

Default

If no path name is given, the remote Server FTP sends status information relative to parameters, connection status.

Usage Guidelines

- The *path_name* argument is optional.
- When **REMSTAT** is issued between data transfer operations, the *path_name* argument can be given.
- With **REMSTAT**, the information is transferred over the control connection instead of the data connection.
- The syntax of *path_name* depends on the associated Server FTP

The TCPaccess Server FTP program implements some additional parameters on the **remstat** command. Use a **help REMSTAT** Client FTP2 command to find additional parameters.

Example

```
REMSTAT /u/lpn
502 STAT command not implemented.
```

RENAME

The **rename** command directs a remote Server FTP to rename a file.

RENAME [*old_path_name*] [*new_path_name* **]**

Syntax Description

old_path_name	File name to be renamed.
new_path_name	New name to be assigned to that file.

Default

If you omit either argument, you are prompted to enter it.

Usage Guidelines

The syntax of the path names depends on the associated Server FTP.

Example

```
RENAME titlecol coltitle
350 File exists, ready for destination name
250 RNTO command successful.
```

REST

The **rest** (restart) command shows the Server FTP the restart marker where a file transfer is to be restarted. This command does not cause a file transfer but instead causes the Server FTP to skip over the file to the specified data checkpoint. This command should be followed immediately by the Client FTP2 command that causes the file transfer to resume.

Note: The restart facility requires that you run in Mode B. Very few UNIX implementations support Mode B and these are unable to use the restart facility.

REST *local_marker remote_marker*

Syntax Description

local_marker	Local marker where the restart is to begin.
remote_marker	Remote marker from which the restart is to begin.

Usage Guidelines

- Both marker arguments are required and represent the Server FTP marker where the file transfer is to be restarted.
- The format of the restart marker is determined by the sending Server FTP and should be entered exactly as displayed during the interrupted file transfer.

Example

The following example shows a restart marker message received by a user during a previous file transfer:

```
B:110 MARK 010023010E8C = 010023010E8C
```

To restart the file transfer at the restart markers, issue the commands shown in bold in this output display:

```
MODE B
BINARY
SITE REST(100000)
REST 010023010E8C 010023010E8C
PUT 'scm.p016572.T01TCP' job16572.MYFILE
350 Requested file action pending further information
350 Requested file action pending further information
150-Data set open with attributes:
Type I N   Stru F   Mode B   Recall 5   Path USER1.JOB16572.MYFILE
Volser ICSPK2   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 133
Blksize 6650   Space 3 1 Cyl Rlse   Restart at 010023010E8C
-Data set open with attributes:
Type I N   Stru F   Mode B   Path SCM.P016572.MYFILE
Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 133
Blksize 6650  Rlse  Bytes/Restart 100000   Restart at 010023010E8C
110 MARK 010025040F96 = 010025040F96
110 MARK 0100280110A0 = 0100280110A0
-Transfer complete
 223966 bytes sent in 1.91 seconds (117259 bytes/s)
Path SCM.P016572.MYFILE   User COLMBIA   Data bytes sent 218918
Disk tracks read 6   Restart markers send 2
```

```
226-Transfer Complete
 223966 bytes received in 1.90 seconds (117876 bytes/s)
Path USER1.JOB16572.MYFILE   User USER1   Data bytes received 218918
Disk tracks written 6   Records folded 1646
Restart markers received 2
```

RMDIR

The **rmdir** (remove directory) command directs a remote Server FTP to remove the specified directory.

RMDIR [*path_name*]

Syntax Description

path_name Directory to be removed.

Default

If you omit *path_name*, you are prompted for it.

Usage Guidelines

- If the path name is relative, the specified subdirectory is removed from the current working directory.
- If the path name is absolute, the specified directory is removed.
- The syntax of *path_name* depends on the associated Server FTP.

Note: Many systems require the directory to be empty before it can be removed.

Example

As an example, if a UNIX Server FTP in session with the Client FTP2 program has */u/user1/work* as the current directory, and a **rmdir** *junk* command is issued by Client FTP2 to that UNIX Server FTP, the *junk* subdirectory of the current directory is removed. The same result is achieved by specifying **rmdir** */u/user1/work/junk*.

```
RMDIR /u/lpn/d.samp
250 RMD command successful.
```

SEND

The **SEND** command is the same as the **PUT** command. Read “PUT” on page 2-51 for details about the **SEND** command.

SEND [*local_path*] [*remote_path*]

Syntax Description

local_path	File name of the file to be copied from the local server.
remote_path	File name of the file at the remote Server into which the file from the local server is copied.

Default

If you omit either path, you are prompted for the file name.

Usage Guidelines

- The command name **PUT** can be used in place of **SEND**. There are no other differences when using **PUT**.
- The syntax for each path depends on the associated Server FTP.

Example

```
SEND n.d temp
-Data set open with attributes:
Type I N   Stru F   Mode B   Path USER1.N.D
Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 3120   Rlse   Bytes/Restart 500000
150 ASCII data connection for temp (138.42.224.15,4115).
-Transfer complete
 6726 bytes sent in 1.58 seconds (4256 bytes/s)   Path USER1.N.D
User COLMBIA   Data bytes sent 6480
Disk tracks read 1
226 ASCII Transfer complete.
```

SITE

The **SITE** (site parameters) command provides the local Server FTP with specific information it requires. This information is essential to file transfers involving that Server FTP, but is not sufficiently universal to have been included specifically in the FTP. Typically, you use a **HELP SITE** Client FTP2 command to find the **SITE** requirements for a specific local Server FTP. Otherwise, review the Server FTP documentation for the **SITE** requirements.

SITE *parameters*

Syntax Description

SITE command parameters are described in “SITE” on page 5-15

Example

This is an example of a **SITE** command to change the default FTP volume:

SITE vol(mvsts2)

SNDS

The **SNDS** (resend **SITE** parameters) command directs the Client FTP program to resend the last **SITE** command to the local Server FTP. Your **SITE** command is reissued without your having to retype it. Since most Server FTPs require that new site parameters be provided before each data transfer, the **SNDS** saves time if identical site parameters are to be used repeatedly.

SND

Example

```
SNDS
SITE vol(mvsts2)
SNDS
site vol(mvsts2) <SENT
```

STATUS

The **STATUS** command requests a status response from the local system.

STATUS [*parameter*]

Syntax Description

parameter	Specifies the directory to be created.

Defaults

- If no parameter is given, the indicated local Server FTP sends status information relative to parameters, such as connection status.
- If no data connection is active, the port is zero.

Usage Guidelines

- The *parameter* argument is optional.
- When the **STATUS** command is issued, the information is transferred over the control connection instead of the data connection.

Example

The TCPaccess Server FTP program implements some additional parameters on the **STATUS** command. Use a **HELP STAT** Client FTP2 command to find additional parameters.

STATUS

```
Connected to: UNIX
--- STATUS ---
-- FTP Parameters --
Remote DT Host, Port 138.42,32,160, 0
Local  DT Host, Port 138.42.224.15, 0
Type I N   Stru F   Mode B   Recall 5   Server is passive
-- END --
-- Control --
User COLMBIA   Acct Accs E0000200   Unit SYSALLDA   Host 138.42.224.15
-- End Control --
-- Path Data --
Volser ICSPK2   Rlse
-- End Path Data --
-- Transfer Information --
Data transfer not in progress   Data bytes sent 6480
Disk tracks read 1   Network bytes sent 6726   Elapsed time 00.00.01
Bytes/Second 4256
-- END --
211 <End of Status>
```

STRUCT

The **STRUCT** (file structure) command provides information on file structure to the remote Server FTP.

STRUCT F | R

Syntax Description

F	File structure is specified by **F**. This is the default if no **STRUCT** command has been used. File structure is used for files with no internal structure, and the file is considered to be a contiguous sequence of data bytes.
R	Record structure is set by **R**. This is for files made up of sequential records. Record structure is accepted for text files (i.e., files with type ASCII or EBCDIC) by all FTP implementations.

Usage Guidelines

One argument is required on the **STRUCT** command. The argument sets the file structure.

Example

```
STRUCT  F
```

SUNIQUE

Use the **SUNIQUE (STORE UNIQUE)** command to store transferred files by unique file names on a remote machine. **SUNIQUE** is a toggle command (meaning it is either off or on). When used, the target server automatically ensures that files received in FTP transfer are stored under a unique name.

SUNIQUE

Syntax Description

This command has no arguments or keywords.

Default

Default for **SUNIQUE** is **OFF**.

Usage Guidelines

- The target remote FTP server must support the **STOU** (store unique) command.
- If **SUNIQUE** is not used, FTP2 will issue the standard **STOR** command. If file names are not unique, files in the target directory could be overwritten.

TYPE

The **TYPE** command tells a Server FTP the data type to use.

TYPE I | **L** *byte_size* | { **A** | **E** [**N** | **T** | **C**] }

Syntax Description

I	Indicates image type. The data is sent as a contiguous bit stream that, for transfer, is packed into 8-bit transfer bytes. The receiving site stores the data as contiguous bits. The receiving storage system might need to pad the file (or each record, in record-structured files) to some convenient boundary. Review the documentation for a Server FTP to find out about padding. Image type is for the efficient storage and retrieval of files and for transfer of binary data. All FTP implementations are required to support the image type.
L *byte_size*	Indicates the local file type and the logical byte size of the file. The byte size value (*byte_size*), representing the logical byte size, is required with the local type. With this type, the data is transferred in logical bytes of the specified size. The logical byte size might differ from the transfer byte size. If the logical and transfer byte sizes differ, the logical bytes are packed contiguously disregarding transfer byte boundaries and are padded at the end if necessary. When the data reaches the receiving host, it is transformed in a manner dependent on the logical byte size and the particular host. The transformation is invertible; an identical file can be retrieved if the same parameters are used. The local type is set by **L**. { **A** | **E** [**N** | **T** | **C**] }
A	Sets the file type to ASCII. This type is accepted by all FTP implementations and is good for transferring text files, except when both hosts find the EBCDIC type more convenient. In accordance with the *NVT* standard, the CRLF sequence is used at the end of a line of text. The sender converts the data from an internal character representation to the standard 8-bit NVT ASCII representation (see the Telnet specification in the list of reference documents). The receiver converts the data from this standard form to the receiver's own internal form.
E	Sets the files type to EBCDIC. This type performs efficient transfer between hosts that use EBCDIC. TCPaccess Client FTP2 users usually use this type when copying files to their MVS host. For transmission, data is 8-bit EBCDIC characters. The character code is the only difference between EBCDIC and ASCII types. End-of-line is rarely used with EBCDIC type to denote structure, but where it is necessary, the NL character is used. The types ASCII and EBCDIC optionally take a second parameter that indicates what kind of vertical format control, if any, is associated with a file. If a file is to be sent to a host for printing, the receiving host must know how the vertical format control is represented. Therefore, the ASCII and EBCDIC types have a second parameter specifying non-print, Telnet, or carriage control (ASA). These are the vertical format control specification options:
N	Sets non-print format control. This is used when the file does not contain vertical format information. Normally, this format is used with files destined for processing or for storage. Non-print format is accepted by all FTP implementations.
T	Sets Telnet format control. This is used when the file contains ASCII/EBCDIC vertical format controls (i.e., CR, LF, NL, VT, FF). The characters CRLF, in exactly this sequence, also denote end-of-line.

C Sets carriage control (ASA) format control. This is used when the file contains ASA (FORTRAN) vertical format control characters.

ASA standard specifies these control characters:

Defaults

- **A** is the default argument for the **TYPE** command.
- ASCII is the default file type.
- For both ASCII and EBCDIC file types, vertical format control N is the default.

Usage Guidelines

- One of the four arguments (I, L *byte_size*, A, or E) is required.
- If local type (**L**) is set, the integer byte size argument must also be set.
- If ASCII (A) or EBCDIC (E) type is set, one of the three vertical format control arguments, N, T, or C can also can be set.

Example

blank	Move paper up one line.
0	Move paper up two lines.
-	Move paper up three lines.
1	Move paper to top of next page.
+	No movement (i.e., overprint).

Restart Support

If a file transfer is interrupted, it can be restarted. However, restart support requires that mode B be specified.

The restart marker provided by TCPaccess is six bytes long in the format *VTTRBB*. **V** is the volume sequence number, *TTR* is the standard IBM OS disk block address, and *BB* is a byte offset within the block.

Using Restart

Use the **SITE REST(***XXXXX***)** Client FTP2 command to tell a TCPaccess FTP Server how often to send a restart marker. Send a restart marker after the sending of the record that exceeds or equals *XXXXX* number of bytes (varying between 1 and 500,000). Send the **SITE** command only to the **RETR** side of a data transfer. Other FTP Servers may initiate sending restart markers in a different way from TCPaccess.

A 110 message is written once per output block if a restart marker is sent somewhere in the data written for the block.

For example, suppose during a data transfer this restart mark message is sent:

```
110 MARK 0100030212C0 = 010003020FA0
```

You can restart the transfer at this point by sending the first number to the **RETR** side and the second number to the **STOR**. If you receive this restart mark message during an aborted data transfer, you can restart the transfer at these disk locations with these Client FTP2 **REST** commands:

Send this command to the RETR side:

REST 0100030212C0

Send this command to the STOR side:

REST 010003020FA0

TCPaccess supports restart markers (set at default value of every 32767 data bytes) if these conditions exist:

- TYPE I
- MODE B

Client FTP2 File Transfer Examples

This section provides some examples of file transfers.

Example Notes

These notes clarify the Client FTP2 examples included in this section:

- Issue **FTP2** when under TSO to enter Client FTP2.
- Text can be entered in uppercase or lowercase. Some host systems support a mixture of lowercase and uppercase letters, while other host systems use uppercase for most functions. All commands entered are translated to uppercase before being sent to the servers. The data associated with a command is sent to its appropriate FTP server without case translation. The TCPaccess FTP server translates user IDs, passwords, data set names, and similar items to uppercase before the commands associated with them are executed.
- An **OPEN** command attempts to connect to a remote host. FTP2 attempts to log the user on to the remote host once the connection is made.

The examples are shown in the following sections.

PUT Example

In the following example FTP2 session, the **PUT** command transfers a file from local MVS host to remote host unix.

```
Setting local directory to the TSO profile prefix
250 “USER1.” is current prefix
TCPaccess Release 5.2 Client FTP2 - Enter command or’?’
open unix
220 unix FTP server (SunOS 4.1) ready.
Enter name (UNIX:user1sys1): user_name
331 Enter PASS command
Password: passwd
230 User user1 loggged in.
put cntl(iefbr14) jclbr14
-Data set open with attributes:
Type A N  Tabs 8  Stru F  Mode S  Path USER1.CNTL(IEFBR14)
Volser COLPK1  Unit SYSALLDA  Dsorg PO  Recfm FB  Lrecl 80  Blksize 3120  Rlse
150 ASCII data connection for jclbr14 (138.42.224.15,4120).
-Transfer complete
820 bytes sent in 0.26 seconds (3153 bytes/s)
Path USER1.CNTL(IEFBR14)
User USER1 Data bytes sent 800
Disk tracks read 1
226 ASCII Transfer complete.
end
```

- FTP2 makes a connection to the MVS local host. The connection is established automatically unless the NOA invocation option was specified.

  User USER1SYS1 is logged on to the local host using information gathered in the TSO address space. FTP2 discovered that user USER1SYS1 has a different TSO profile prefix than his user ID. FTP2 set the local working directory to his current TSO profile prefix of USER1.

- The **open unix** command establishes a connection to the remote UNIX host. FTP2 automatically prompts for a user ID, password, and optionally, an account.
- At the “Enter name...” entry, FTP2 prompts you for the correct user ID on UNIX.

  User USER1SYS1 can press **Enter** and get a default user ID of user1sys1, or can override the user ID and enter a different user ID.

  A user ID of user1 was entered. A password is prompted for and entered in a non-display field. The user ID and password are valid and user user1 logs on to the remote host.

- The **put cntl(iefbr14) jclbr14** command tells FTP to transfer the file USER1.CNTL(IEFBR14) from the local host (MVS) and create or overwrite file jclbr14 on the remote host, unix.

  The 226 message indicates a successful transfer occurred. Any 400 or 500 level message indicates a data transfer failure.

- The **end** command ends the FTP2 session.

GET Example

In the following example FTP2 session, the **GET** command transfers a file from remote host unix to the local MVS host.

```
TCPaccess Rn Client FTP2 - Enter command or '?'
open unix
220 unix FTP server (SunOS 4.1) ready.
Enter name (UNIX:user1): user_name
331 Enter PASS command
Password: passwd
230 User user1 logged in.
get jclbr14 cntl(newbr14)
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5
Path USER1.CNTL(NEWBR14)
Volser COLPK1   Unit SYSALLDA   Dsorg PO   Recfm FB   Lrecl 80
Blksize 3120   Space 31 15 Tracks Rlse
150 ASCII data connection for jclbr14 (138.42.224.15,4121) (810 bytes).
-Transfer complete
 820 bytes received in 10.1 seconds (80 bytes/s)
Path USER1.CNTL(NEWBR14)   User USER1   Data bytes received 800
Disk tracks written 1
226 ASCII Transfer complete.
end
```

- The **open unix** command establishes a connection to the remote host **unix**.

  A connection is made to the MVS local host. **user1** is logged on to the local host using information gathered in the TSO address space.

  After establishing a connection, FTP2 automatically prompts for a user ID, password, and, optionally, an account.

- At the "Enter name..." entry, FTP2 prompts user1 for the correct user ID on unix. At this prompt, user1 can press **Enter** and get a default user ID of user1, or can override the user ID and enter a different user ID.

  In this example, **user1** just hit the enter key, so the default user ID is used. A password is prompted for and entered in a non-display field. The user ID and password are valid and **user1** logs on to the remote host.

- The command **get jclbr14 cntl(jclbr14)** tells FTP to transfer the file jclbr14 on remote host unix to file USER1.CNTL(NEWBR14) on the local MVS host.

  The 226 message indicates a successful transfer. Any 400 or 500 level message indicates a data transfer failure.

- The **End** command ends the FTP2 session.

FTP2 Restart Marker Example

When transferring large amounts of data across many channels, the transfer sometimes does not complete successfully. The following example FTP2 session shows how to use the **rest** (restart) command to imbed marks to tell Server FTP where to restart a file transfer. Read "REST" on page 2-61.

```
TCPaccess Rn - Usr FTP2 - Enter command or '?'
open mvs
220 MVS.HQ.COMPANY.COM -- FTP2 Server, Enter command or HELP
Enter name (MVS:columbia): user1
331 Enter PASS command
Password: passwd
230 LOGGED IN - HOST 138.42.128.13 USER user1
binary
mode b
site rest(100000)
remsite space(3 1) cyl recfm(vb) blk(6650) lrecl(133)
put 'scm.p016572.T01TCP' psr16572.job
-Dataset open with attributes:
Type I N   Stru F   Mode B   Path SCM.P016572.T01TCP
Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 133
Blksize 6650 Rlse Bytes/Restart 100000
150-Dataset open with attributes:
Type I N   Stru F   Mode B   Recall 5   Path USER1.PSR16572.JOB
Volser HAGCAT   Unit SYSALLDA   Dsort PS   Recfm VBV   Lrecl 133
Blksize 6650 Space 3 1 Cyl Rlse
-Transfer complete
 1656466 bytes sent in 13.5 seconds (122338 bytes/s)
Path SCM.P016472.T01TCP   User COLMBIA   Data bytes sent 1619142
Disk tracks read 41   Restart markers sent 16
110    MARK    000204010a      = 00020414F9
110    MARK    0005010214      = 0005021075
110    MARK    000704031E      = 0007060BF1
110    MARK    000A010428      = 000A040771
110    MARK    000C040532      = 000D0202ED
110    MARK    000F01063C      = 000F0517DE
110    MARK    0011040746      = 001203135E
110    MARK    0014010850      = 0015010EDA
110    MARK    001604095A      = 0017050A56
110    MARK    0019010A64      = 001A0305D6
110    MARK    001B040B6E      = 001D010152
110    MARK    001D010C78      = 001F041643
110    MARK    0020040D82      = 00220211C3
110    MARK    0020040D82      = 00220211C3
110    MARK    0023010E8C      = 0024060D3F
110    MARK    0025040F96      = 00270408BB
110    MARK    00280110A0      = 002A02043B
226-Transfer complete
1656466 bytes received in 13.6 seconds (120998 bytes/s)
Path USER1.PSR16572.JOB  User USER1   Data bytes received 1619142
Disk tracks written 43              Records folded 12551
Restart markers received 16
```

- FTP2 starts up and automatically connects and signs on to the local host. Then the user connects and signs on to the remote host mvs.
- The “binary” entry sets the data type to binary.
- The “mode b” entry invokes block mode, which tells the FTP servers that restart markers are being used.
- The “site rest(100000)” entry tells the local FTP server how often to place restart markers into the data.
- The **remsite** command issued to the remote site allocates a data set with the correct attributes.
- The file SCM.PO16572.MYFILE is transferred from the local MVS host to host mvs as USER1PSR16572.JOB. All the data was successfully transferred.

  If the data transfer fails you can restart the data transfer using any of the “110 MARK ...” restart markers.

  Read “REST” on page 2-61 for more details.

Transfer to an MVS Internal Reader

The following example FTP2 session shows an FTP2 file transfer from a file on host unix to an MVS internal reader on the local MVS host.

```
TCPaccessRn Client FTP2 - Enter command or '?'
open unix
220 unix FTP server (SunOS 4.1) ready.
Enter name (UNIX:user1): user_name
331 Enter PASS command
Password: passwd
230 User user1 logged in.
site submit
get jclbr14 anyname
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Intrdr   Recfm FB   Lrecl 80   Blksize
20000
150 ASCII data connection for jclbr14   (138.42.224.15,4122) (810 bytes).
-Transfer complete
 820 bytes received in 4.17 seconds (196 bytes/s)   User USER1
Data bytes received 800
226 ASCII Transfer complete.
end
```

- The **open unix** command establishes a connection to the remote host **unix**. The user logs on to the remote host.
- The command **site submit** is a command to the local MVS host directing the next data transfer to the MVS internal reader for execution. **SITE** commands are relevant to an MVS host where TCPaccess is running.
- The command **get jclbr14 anyname** commands FTP2 to transfer file jclbr14 from remote host unix. Due to the previous **SITE** command, the file is transferred to an MVS internal reader on the local MVS host. The file name (anyname) for the local host is ignored because no data set is being created or updated.
- The **end** command ends the FTP2 session.

Managing Directories on UNIX-based Systems

The following example FTP2 session shows the directory manipulation commands for the remote host.

```
TCPaccess Rn Client FTP2 - Enter command or '?'
open unix
220 unix FTP2 server (SunOS 4.1) ready.
Enter name (UNIX:user1): user_name
331 Enter PASS command
Password: passwd
230 User user1 logged in.
pwd
257 "/home/unix/user1" is current directory.
mkdir tempdir
A:257 MKD command successful.
pwd
257 "/home/unix/user1" is current directory.
cwd tempdir
250 CWD command successful.
pwd
257 "/home/unix/user1/tempdir" is current directory.
put cntl(iefbr14) jclbr14
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Path USER1.CNTL(IEFBR14)
Volser COLPK1   Unit SYSALLDA   Dsorg PO   Recfm FB   Lrecl  80
Blksize 3120   Rlse
150 ASCII data connection for jclbr14 (138.42.224.15,4123).
-Transfer complete
 820 bytes sent in 0.54 seconds (1518 bytes/s) Path USER1.CNTL(IEFBR14)
User USER1   Data bytes sent 800
Disk tracks read 1
226 ASCII Transfer complete.
dir
-Data set open with attributes:
Type A N   Tabs 8   Stru F   Mode S   Recall 5
Path USER1.FTP.TMP.T2334662
Volser ICSUSR   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 80
Blksize 6160   Space
5 3 Tracks Rlse
150 ASCII data connection for /bin/ls (138.42.224.15,4124) (0 bytes).
226 ASCII Transfer complete.
-Transfer complete
 72 bytes received in 9.29 seconds (7 bytse/s)
Path USER1.FTP.TMP.T2334662   User COLMBIA   Data bytes received 68
Disk tracks written 1   Records padded 2
total 1
-rw-rw-rw-  1 user1   dvlp   810 Jan 26 10:19 jclbr14
IDC0550I ENTRY (A) USER1.FTP.TMP.T2334662 DELETED
dele jclbr14
250 DELE command successful.
cdup
250 CWD command successful.
pwd
```

```
257 "/home/unix/user1" is current directory.
rmd tempdir
250 RMD command successful.
pwd
257 "/home/unix/user1" is current directory.
end
```

- User1 connects and logs on to the local host and remote host unix.
- The **pwd** command asks the server to print the current directory on the remote host.
- The command **mkdir tempdir** asks the remote ftp host server to create a directory called tempdir.
- By creating a directory in the third section of this example, you have not changed your current working directory on the remote host.

  The command **cwd tempdir** changes the remote host directory from /home/unix/user1 to /home/unix/user1/tempdir. The **pwd** command confirms the directory.

- The **put cntl(iefbr14) jclbr14** command copies a file from the local MVS host to the remote host as file jclbr14.
- The **dir** command finds all the current members on the current remote host directory. The only file in the directory is jclbr14.
- The **dele jclbr14** command removes the file from the remote host.
- The command **cdup** changes the remote host directory from its current directory /home/unix/user1/tempdir to its parent directory home/unix/user1. The **pwd** command reflects this change.
- The command **rmd tempdir** asks the remote host server to remove directory /home/unix/user1/tempdir.
- The **end** command terminates the FTP2 session.

Transferring and Using a File in a Single JCL Job

Transferring a file in one job step and using that file in another job step can run into a file allocation problem, wherein the file transfer fails in the FTP2 job step. This is usually because the file has been allocated to the batch job and the TCPaccess base product cannot allocate the file for a data transfer. The FTP2 and FTP client programs do not perform file transfers in their own address space; the FTP2 and FTP clients direct the TCPaccess base product to perform file transfers.

As a workaround, perform an IDCAMS ALTER NEWNAME of the file between the file transfer step and the use of the file.

Examples

The following sample JCL does a file transfer, performs an IDCAMS ALTER NEWNAME, and uses the file in the final job step (the data set MVS.P25206.DATA is created in the first job step, renamed to the file MVS.NEWNAME.DATA in the second step, and the file name MVS.NEWNAME.DATA is used in the last step):

```
//MVSA JOB (TSO00...99),'FTP2 BATCH',MSGCLASS=X,NOTIFY=MVS,CLASS=A
//*
//*  JOB TO TRANSFER A FILE AND THEN USE IT IN A LATER JOBSTEP
//*
//*  STEP TO DO THE FTP TRANSFER
//*
//FTP1        EXEC PGM=IKJEFT01,REGION=4000K
//SYSTSIN     DD *,DCB=BLKSIZE=80
    FTP2 / APP=ACCES  TEST NETRC
//STEPLIB     DD DCB=BLKSIZE=32000,
//               DISP=SHR,DSN=T01TCP.LINK
//SYSTSPRT    DD SYSOUT=X
//SYSPRINT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPUT      DD SYSOUT=*,DCB=BLKSIZE=133
//SYSVLT      DD SYSOUT=*,DCB=BLKSIZE=133
//NETRC       DD DISP=SHR,DSN=MVS.FTP.NETRC
//SYSGET      DD *,DCB=BLKSIZE=80
open unix
get temp 'mvs.p25206.data'
END
/*
//*
//*   STEP TO PERFORM AN IDCAMS ALTER NEWNAME
//*
//STEP2       EXEC PGM=IDCAMS
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
 ALTER 'MVS.P25206.DATA'                NEWNAME('MVS.NEWNAME.DATA') -
  CAT(CATALOG.TSO.VESA001)
//*
//*   THIS STEP PRINTS THE FILE ON THE SYSUT1 DD CARD
//*
//STEP3       EXEC PGM=IEBGENER
//SYSUT1      DD DISP=SHR,DSN=MVS.NEWNAME.DATA
//SYSUT2      DD SYSOUT=X,COPIES=1,DCB=*.SYSUT1
//SYSPRINT    DD   SYSOUT=*
//SYSIN       DD   DUMMY
```

The following sample JCL uses a file, performs an IDCAMS ALTER NEWNAME, and does a file transfer in the final job step (the file MVS.OLDNAME.DATA is used in the first jobstep, renamed to the file MVS.TRANSFER.DATA in the second jobstep, and the file MVS.TRANSFER.DATA is transferred in the last jobstep):

```
//MVSB JOB (TSO00,,,99),'FTP2 BATCH',MSGCLASS=X,NOTIFY=MVS,CLASS=A
//*
//* JOB TO TRANSFER A FILE AND USE IT IN A LATER JOBSTEP
```

```
//*
//*   THIS STEP PRINTS THE FILE ON THE SYSUT1 DD CARD
//*
//STEP1     EXEC PGM=IEBGENER
//SYSUT1    DD DISP=SHR,DSN=MVS.OLDNAME.DATA
//SYSUT2    DD SYSOUT=X,COPIES=1,DCB=*.SYSUT1
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    DUMMY
//*
//*   STEP TO PERFORM AN IDCAMS ALTER NEWNAME
//*
//STEP2     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 ALTER 'MVS.OLDNAME.DATA'                NEWNAME('MVS.TRANSFER.DATA') -
  CAT(CATALOG.TSO.VESA001)
//*
//*  STEP TO DO THE FTP TRANSFER
//*
//FTP3      EXEC PGM=IKJEFT01,REGION=4000K
//SYSTSIN   DD *,DCB=BLKSIZE=80
FTP2 / APP=ACCES  TEST NETRC
//STEPLIB   DD DCB=BLKSIZE=32000,
//          DISP=SHR,DSN=T01TCP.LOAD
//SYSTSPRT  DD SYSOUT=X
//SYSPRINT  DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPUT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSVLT    DD SYSOUT=*,DCB=BLKSIZE=133
//NETRC     DD DISP=SHR,DSN=MVS.FTP.NETRC
//SYSGET    DD *,DCB=BLKSIZE=80
open unix
put 'mvs.transfer.data' temp
END
/*
```

3

Client FTP3

This chapter describes Client FTP3, the File Transfer Protocol (FTP) that allows file transfers among unlike hosts in diverse internetworking environments. It contains these sections:

- “Introducing Client FTP3” on page 3-2
  Provides a brief overview of the TCPaccess File Transfer Protocol implementation.
- “Client FTP3” on page 3-3
  Describes the FTP3 data flow and includes an illustration showing how Client FTP3 works.
- “Invoking Client FTP3” on page 3-4
  Describes how to use Client FTP3 as both a TSO command and as a regular batch program.
- “General Client FTP3 Operation” on page 3-17
  Describes the general operation of the Client FTP3 program.
- “Client FTP3 Commands” on page 3-22
  Includes a table listing all of the Client FTP3 commands and provides a brief description of each. Commands are:

“? Command” on page 3-24	“ACCOUNT” on page 3-25	“APPEND” on page 3-26	“ASCII” on page 3-27	“BINARY” on page 3-28
“CD” on page 3-29	“CDUP” on page 3-30	“CLOSE” on page 3-31	“DEBUG” on page 3-32	“DELETE” on page 3-33
“DELIMIT” on page 3-34	“DIR” on page 3-35	“EBCDIC” on page 3-36	“GET” on page 3-37	“HELP” on page 3-38
“LCD” on page 3-39	“LMKDIR” on page 3-40	“LOCSITE” on page 3-41	“LOCSTAT” on page 3-42	“LPWD” on page 3-43
“LS” on page 3-44	“MDELETE” on page 3-45	“MGET” on page 3-46	“MKDIR” on page 3-48	“MODE” on page 3-49
“MPUT” on page 3-51	“NOOP” on page 3-52	“OPEN” on page 3-53	“PASS” on page 3-54	“PUT” on page 3-55
“PWD” on page 3-56	“QUIT” on page 3-57	“QUOTE” on page 3-58	“RENAME” on page 3-59	“RESTART” on page 3-60
“RMDIR” on page 3-61	“SENDSITE” on page 3-62	“SITE” on page 3-63	“STATUS” on page 3-64	“STRUCT” on page 3-65
“SYSTEM” on page 3-66	“TRACE” on page 3-67	“TSO” on page 3-68	“TYPE” on page 3-69	“USER” on page 3-71

Introducing Client FTP3

Like Client FTP and Client FTP2, Client FTP3 is a three-party model FTP client. Two control connections are established and maintained by the client. Like Client FTP2, Client FTP3 connects automatically to the local TCPaccess FTP server and signs on the user.

Despite these basic similarities, Client FTP3 differs from both Client FTP and Client FTP2 in several significant ways. The differences are described in this chapter.

PL/I Runtime Libraries

Client FTP3 is written in C, and so does not require PL/I runtime libraries, as do the other clients. It does require the SAS/C runtime libraries.

Connections to the FTP Servers

Client FTP3 is a true client application. Both Client FTP and Client FTP2 communicate with a client *agent* through a VTAM LU0 connection in the TCPaccess server address space. Client FTP3, on the other hand, uses the TCPaccess C/Sockets library to establish direct socket connections with both the local TCPaccess FTP server (local server) and the remote FTP server. This eliminates the need for VTAM LU resources, and improves response time.

Throughput and CPU Utilization

Although primarily a three-party client, Client FTP3 performs some operations in two-party mode to take advantage of the high throughput and low CPU utilization of the TCPaccess FTP server. The result is improved response to the user and quick response time of a direct connection to the remote server.

For file-transfer operations, such as the client commands **GET**, **PUT**, and **APPEND**, Client FTP3 works in three-party mode. For directory commands, such as the client commands **LS** and **DIR**, as well as the implied directory commands in the **MGET**, **MPUT**, and **MDELETE** commands, Client FTP3 operates in two-party mode. Unlike Client FTP2, Client FTP3 does not need to be run under the TMP to support **LS** and **DIR** in batch jobs.

Client FTP3 Data Transfer

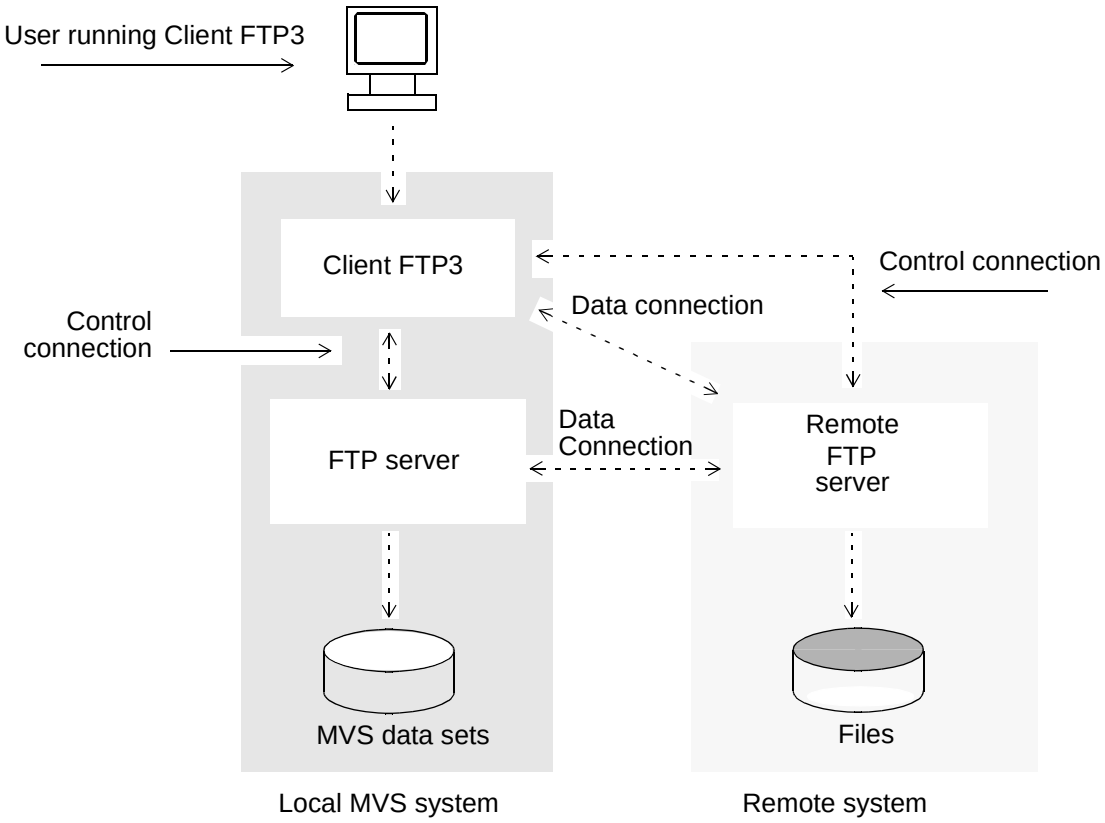
Client FTP3 appears to the user as a two-party model by suppressing almost all local server replies, simplifying the client responses considerably. Also, Client FTP3 operates in *blocked* mode. When a data transfer is initiated, the terminal remains blocked until the transfer completes.

By comparison, Client FTP2 returns a command prompt and allows the user to enter commands as soon as a data transfer is initiated between the local and remote servers. This allows you to enter the **STAT** command to check on progress, or the **ABORT** command to abort the transfer, but it also means that user running interactively must press **ENTER** to be notified of the end of the transfer. Client FTP3, on the other hand, does not allow terminal input during a data transfer. For long-running transfers, a statistics message is written to the terminal every 10 seconds. You can abort the transfer by pressing the attention key on the terminal keyboard. When the transfer ends, Client FTP3 displays the reply, and returns to the command prompt.

Client FTP3

"Figure 3-1" on page 3-3 shows the relationship between the Client FTP3 program and the two Server FTP programs in the three-party model.

Figure 3-1 Client FTP3 Program in the Three-Party Model



Note that two Data Connections are shown. The data connection between the Remote Server FTP and the Local Server FTP is used for the **GET**, **PUT**, and **APPEND** client commands (**RETR**, **STOR**, **APPE** and **STOU** server operations). The data connection between the Client FTP3 and the Remote Server FTP is used for the client **LS** and **DIR** commands and the implied **LS** in the **MGET**, **MPUT**, and **MDELETE** commands (**LIST** and **NLST** server operations).

Invoking Client FTP3

The Client FTP3 program runs as a TSO command and can be called as a regular batch program with MVS JCL.

Invoking Client FTP3 through TSO

In a TSO environment, Client FTP3 can be accessed as a TSO command or it can be called as a program with the TSO **CALL** command. Because Client FTP3 does not use full-screen facilities, it can be used from any type of terminal supported by TSO, including 3270 systems, 3767 systems, and asynchronous ASCII terminals supported by NTO or NPSI.

Note: You must have PROMPT set in your TSO profile for Client FTP3 to work properly in interactive mode.

A left parenthesis "(" separates the remote_host and port_number from the other options.
Example: `ftp unix.company.com (translate standard`

FTP3 TSO Command

Some keywords used in the FTP3 command can be used to override values specified in the TCPIP.DATA data set. Read "Understanding the Configuration Data Sets" on page 3-9 for more information.

Invoke Client FTP3 by entering the FTP3 TSO command in this format.

FTP3 *remote_host* [*port_number*] [*options*]

Syntax Description

FTP3	Invokes the Client FTP3 program.
remote_host	Name of the remote host. Client FTP3 will automatically connect to this host at initialization. This parameter is required. If you do not supply this parameter, you will be prompted for it.
port_number	Port number of the FTP server on the remote host. Default: 21
BLOCK | NOBLOCK	Specifies whether the ftp client will permit user input while a file transfer is in progress. Default: BLOCK
DEBUG	Toggle used to activate or deactivate the debugging option. Use the DEBUG or TRACE options interchangeably. You may include either DBUG or TRACE on the command line, but not both; the second option cancels the first.
EXIT | EXIT=*nn*	Specifies that the client is to terminate in case of an error if the exit_if_error flag is true. Exit=*nn* provides a return code for error conditions.

LOCALHOST *host_name*	Specifies the host name of the local FTP server host. **Override note:** This option will override the HOSTNAME statement in the TCPIP.DATA data set.
SSID *subsystem_id*	Specifies the subsystem id for the TCPaccess API. Default: ACSS
TCP *tcpip*	Specifies the job name of the TCPaccess sockets API address space. If no job name is specified, FTP3 uses the subsystem ID specified in the SSID parameter of the FTP statement. If SSID does not indicate a subsystem ID, ACSS is used**.** **Override note:** This option will override the TCPIPJOBNAME statement in the TCPIP.DATA data set.
TIMEOUT *nn*	Sets the following timeout parameters MyopenTime DconnTime CconnTime InactTime DataCtTime See the discussion “Understanding the Configuration Data Sets” on page 3-9 for the meaning of these timers.
TRACE	Toggle used to activate or deactivate the debugging option. Use the DEBUG or TRACE options interchangeably. You may include either DBUG or TRACE on the command line, but not both; the second option cancels the first.
TRANSLATE *data_set_name*	Specifies the name of a nonstandard translate table. If this parameter is not supplied, FTP will use the translate table in *hlq*.STANDARD.TCPXLBIN (read “TCPIP.DATA” on page 3-13 for an explanation of the *hlq*). If present, this parameter will be used to construct a data set name in the form *user_id.data_set_name*.TCPXLBIN. If this data set does not exist, FTP will attempt to allocate *hlq.data_set_name*.TCPXLBIN.
VERBOSE	Specifies VERBOSE mode. All commands to and replies from the local host will be echoed to the user.

TSO CALL Command

Use the TSO **CALL** command in a TSO environment to call and execute the FTP3 program out of a specific library. This is especially useful at sites that run multiple releases of the product or have test and production versions of the product at different maintenance levels.

CALL 'T01TCP.FTPLOAD(FTP3)' [*' remote_host port options'*]

Syntax Description

'T01TCP.FTPLOAD(FTP3)	Library from which FTP3 will be called .
'options'	Any number of FTP3 invocation options can be included in the **CALL** command. Read "Invoking Client FTP3 through TSO" on page 3-4 for a complete list.

Usage Guidelines

- The data set name, T01TCP.FTPLOAD, might need to be replaced by the appropriate data set name at your installation. Check with your TCPaccess site administrator.
- When options are specified in the command statement, they must be enclosed in single quotes.
- When invoked by the **CALL** command**,** Client FTP3 runs as a program and not as a TSO command.
- You can use the NETRC file with the TSO FTP3 call. Read "The NETRC File" on page 3-19 for more information.

Batch Invocation

The Client FTP3 program can be run in batch as either a program like any other, or as a TSO command by running it under a batch Terminal Monitor Program (TMP).

You can specify a NETRC file in batch mode. Specify a NETRC DD file with the name of your NETRC file.

Batch Program

You can invoke Client FTP3 in batch in a manner similar to any other batch utility program. A sample JCL file is contained in the SAMP data set as T00PFBJB

```
//<jobname> JOB job_stmt_parms
//FTPSTEP EXEC PGM=FTP3,REGION=1024K,
//            PARM='remote_host port options'
//STEPLIB DD DSN=T01TCP.FTPLOAD,DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=userid.TCPIP.DATA
//SYSFTPD  DD DISP=SHR,DSN=userid.FTP.DATA
//SYSTERM  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN DD *
unix
user pass
stat
quit
//
```

Optionally, you may include a NETRC file, as shown here:

```
//NETRC DD DISP=SHR,DSN=userid.NETRC
```

Batch TMP

An example of this JCL is located in the SAMP data set as member T00PFTMP.

```
//<jobname> JOB job_stmt_parms
//*
//*    RUN FTP3 UNDER BATCH TSO
//*
//FTP3    EXEC PGM=IKJEFT01,REGION=4096K,
//STEPLIB        DD DISP=SHR,DSN=T01TCP.FTPLOAD
//               DD DISP=SHR,DSN=T01TCP.LOAD
//SYSTSPRT       DD SYSOUT=*
//OUTPUT         DD SYSOUT=*,DCB=BLKSIZE=133
//SYSTCPD        DD DISP=SHR,DSN=userid.TCPIP.DATA
//SYSFTPD        DD DISP=SHR,DSN=userid.FTP.DATA
//SYSTERM        DD SYSOUT=*
//SYSTSIN        DD *
FTP3 remote_host port options
//INPUT    DD *
unix
user pass
stat
quit
//
```

Understanding the Configuration Data Sets

If you have previously installed the IBM TCP/IP for MVS, the configuration files described here might already exist. TCPaccess provides support for these data sets to allow former IBM customers to run their applications using TCPaccess.

Client FTP3 automatically searches for and dynamically allocates the configuration data sets when you start FTP3. You can define user-specific environment settings for FTP3 by setting parameters in these configuration data sets:

- *hlq*.TCPIP.DATA

TCPIP.DATA defines the TCPaccess system environment on the local host. You can set your own values by creating a *userid***.**TCPIP.DATA data set.

- *hlq*.FTP.DATA

FTP.DATA defines the local **SITE** parameters that are sent to the local host in the form of **SITE** commands. You can set your own values by creating a *userid***.**FTP.DATA data set.

The high-level qualifier is specified in T00PFUM. See “Changing the High-Level Qualifier” on page 3-12 for instructions on changing the default HLQ.

Samples of these files are located in the SAMP data set. The member names are T00FTPDS (FTP.DATA) and T00TCPDS (TCPIP.DATA).

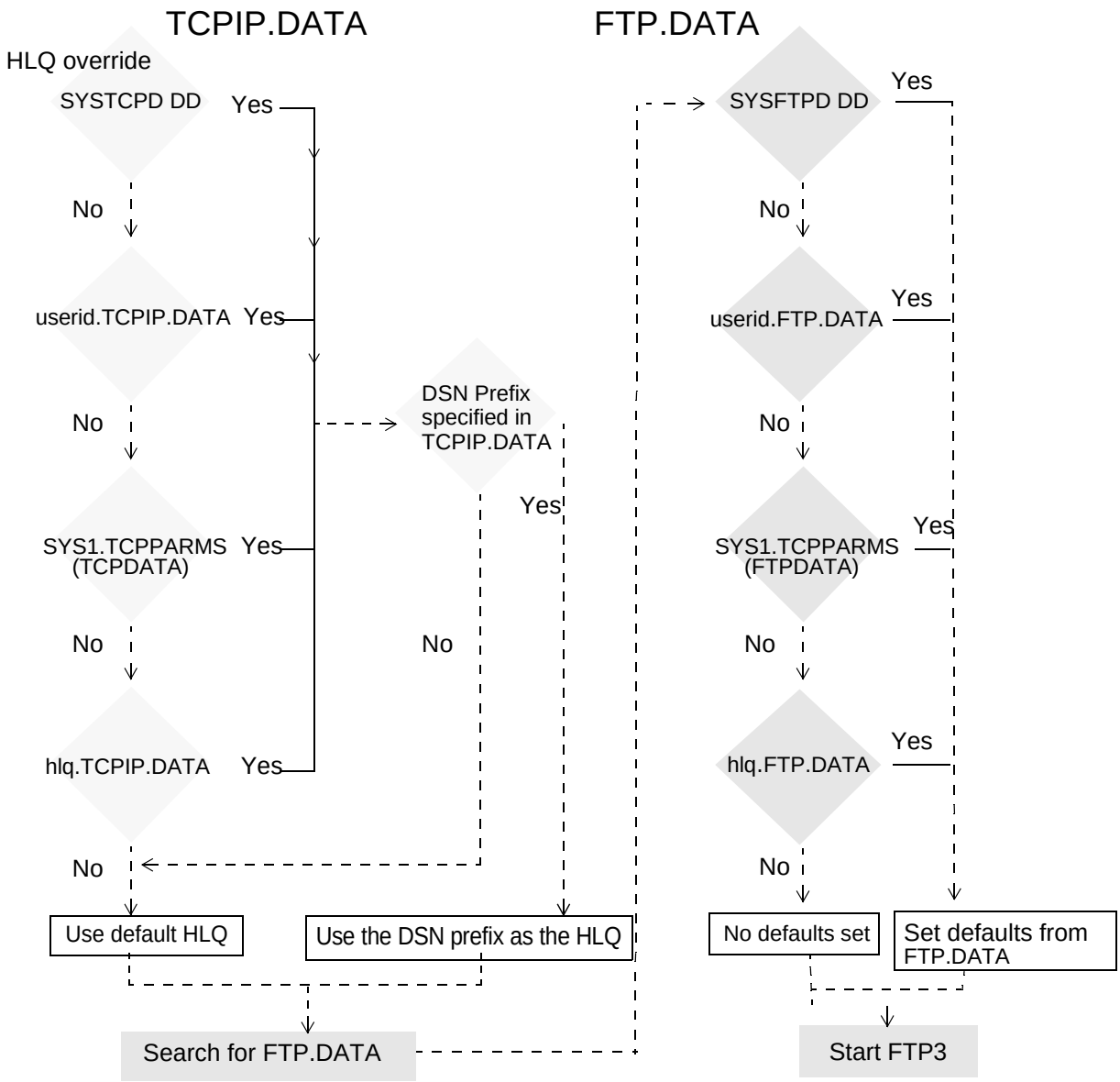
CAUTION ! These configuration data sets must be preconfigured before you execute FTP3.

If you want more details about how the configuration data sets are allocated, see “Dynamic Data Set Allocation” on page 3-10.

Dynamic Data Set Allocation

Client FTP3 searches first for the TCPIP.DATA file. If the TCPIP.DATA file is found and contains a DATASETPREFIX statement, the prefix will be used to search for the FTP.DATA file. You can override the search sequence if you want to use high-level qualifiers other than the defaults for the configuration data sets. If the data sets are not found, the search continues as described for each file. “Figure 3-2” on page 3-10 illustrates the allocation sequence.

Figure 3-2 Dynamic Data Set Allocation (next page)



TCPIP.DATA Allocation

FTP3 follows the search path described below to locate the TCPIP.DATA file.

1. FTP3 looks first for a SYSTCPD DD statement, which is used to define an override for the high-level qualifier.
2. If no SYSTCPD DD statement is located, FTP3 searches for a *userid*.TCPIP.DATA file.

   The user ID is the TSO user ID of the TSO user issuing the FTP3 command or the user ID of the FTP3 batch job if the command is issued from a batch job.

3. If no *userid*.TCPIP.DATA file is located, FTP3 searches for a TCPDATA member in the SYS1.TCPPARMS partitioned data set (PDS).

   SYS1.TCPPARMS will be used during initialization; it may have been created previously if the IBM product was installed.

4. If the TCPDATA member is not located, FTP3 looks for the *hlq*.TCPIP.DATA data set.
5. Once the TCPIP.DATA file is located, FTP3 uses the high-level qualifier defined by the DATASETPREFIX statement in the TCPIP.DATA file.
6. Overriding: If no TCPIP.DATA file is located, no default values are imposed, with the exception that FTP3 will connect to either
   - the resolved subsystem ID of the started procedure name or the job name specified in the TCP option of the FTP3 command

   or

   - the subsystem ID specified in the SSID option in the FTP3 command.
7. If the FTP3 command includes neither the SSID nor TCP option, FTP3 connects to the SSID default subsystem ID, ACSS.

FTP.DATA Allocation

After the TCPIP.DATA search is complete, FTP3 starts searching for the FTP.DATA file. Once the FTP.DATA file is located, the search is complete.

1. It searches first for a SYSFTPD DD statement.

   The data set name specified in this statement will be used by FTP3.

2. If no SYSFTPD DD statement is found, FTP3 searches for a *userid*.FTP.DATA file.

   The user ID prefix in *userid*.FTP.DATA is the TSO user ID of the TSO user issuing the FTP3 command or the user ID of the FTP3 batch job.

3. If no *userid*.FTP.DATA file is located, FTP3 searches for an FTPDATA member in the SYS1.TCPPARMS PDS.

   SYS1.TCPPARMS will be used during initialization; it may have been created previously if the IBM product was installed.

4. If a *userid*.FTP.DATA is not found, FTP3 looks for *hlq*.FTP.DATA.

5. If no FTP.DATA file is located, or if configuration parameters are missing from FTP.DATA, FTP3 sets no defaults except for these timeout parameters:
   - CCONNTIME
   - DATACTTIME
   - DCONNTIME
   - INACTTIME
   - MYOPENTIME

   The default for each of these parameters is 300 seconds.

Normal settings for local SITE parameters are determined by the TCPaccess administrator in the Server FTP configuration statements in APPCFGx*x*.

Related References on the FTP.DATA File

For additional detailed information about the FTP.DATA file, refer to *SC31-7134-01 IBM TCP/IP V3R1 for MVS: Customization and Administration Guide.*

Changing the High-Level Qualifier

Client FTP3 uses both implicit and explicit data set allocation. You can use the pre-assigned data set names, *hlq*. TCPIP.DATA and *hlq*.FTP.DATA, or you can override the pre-assigned data set names with your JCL.

Client FTP3 is distributed with the high-level qualifier TCPIP. The name is defined in the T00PFUM module. You may change the default high-level qualifier by doing one of the following:

- Apply USERMOD member T00PFUM1 in the CNTL data set to change the 26-character field in T00PFUM to a value suitable for your site. The APAR contains instructions for changing the high-level qualifier.
- Specify the DATASETPREFIX statement in the TCPIP.DATA data set at execution time.

TCPIP.DATA

The data set TCPIP.DATA defines TCP/IP parameters. An example of this file is located in the SAMP data set member T00TCPDS.

FTP3 does not support multiple host definitions in a single TCPIP.DATA file. If you have a TCPIP.DATA file which contains more than one host definition, you need to modify your file and create a separate TCPIP.DATA file for each host.

"Table 3-1" on page 3-13 describes the parameters are supported in the TCPIP.DATA data set.

Table 3-1 TCPIP.DATA Parameters

Parameter	Description
DATASETPREFIX *dsn_prefix*	*dsn_prefix* identifies the high-level qualifier to be used to determine the *hlq*.FTP.DATA data set name and the translate table data set name. Default: None
HOSTNAME *host_name*	*host_name* specifies the host name of the local FTP server. Default: None
DOMAINORIGIN *origin*	The *origin* is to be appended to the *host_name* to form the fully-qualified host name. Default: None
MESSAGECASE MIXED | UPPER	Specifies whether messages from the FTP client will be displayed in mixed case or translated to upper case. Default: **MIXED**
TCPIPJOBNAME *tcpip_proc*	Identifies the started procedure name or job name of the TCPaccess stack. This is resolved to a subsystem ID by the client. Default: None

FTP.DATA

The FTP.DATA data set defines local SITE parameters that are sent to the local host in the form of **SITE** commands. You can set your own values by creating a *userid*.FTP.DATA data set. An example of this file is contained in the SAMP data set member T00FTPDS.

The parameters listed in "Table 3-2" on page 3-14 are set in the FTP.DATA file. Many of these parameters correspond to **SITE** commands available in TCPaccess Server FTP . You can use the **LOCSITE** command to change any of these parameters during the FTP session.

Table 3-2 FTP.DATA Parameters

Parameter	Description
AUTOMOUNT TRUE | FALSE	If TRUE, the client will send **SITE AUTOMount** to local host; if FALSE, the client will send **SITE NOAUTOmount**. Default: None.
AUTORECALL TRUE | FALSE	If TRUE, the client will send **SITE AUTORecall** to local host; if FALSE, the client will send **SITE NOAUTOrecall**. Default: None.
BLKSIZE *blk_size* **BLOCKSIZE** *blk_size*	The client will send a **SITE BLKSize=blk_size** command to the local host. Default: None.
CHKPTint *checkpoint_interval*	The client will send a **SITE CHKPTint=checkpoint_interval** command to the local host. Default: None.
CCONNTIME *close_connection_time*	Set internal timer. Defines the timeout value (in seconds) for closing a control connection. Default: 300 seconds.
DATACLASS *SMS_data_class*	Sends a **SITE DATAClass=SMS_data_class** command to the local host. Default: None.
DATACTTIME *data_connection_timeout_time*	Sets internal timer; sends **SITE DIDle=data_connection_timeout_time** to the local host. Defines the timeout value when sending or receiving data. Default: 300 seconds.
DCBDSN *DCB_dataset_name*	Sends a **SITE DCBdsn=DCB_dataset_name** command to the local host. Default: None.
DCONNTIME *data_connection_close_time*	Sets internal timer; sends a **SITE DClose=data_connection_close_time** command to the local host. Defines the timeout value when closing a data connection. Default: 300 seconds.
DIRECTORY *directory_blocks*	Sends a **SITE Directory=directory_blocks** command to the local host. Default: None.

DIRECTORYMODE TRUE | FALSE	If TRUE, the client will send **SITE DIRECTORYmode** to local host; if FALSE, the client will send **SITE DATASETmode**. Default: None.
FILETYPE SEQ | JES	Sends a **SITE FILEType=SEQ | JES** command to the local host. Default: None.
INACTTIME *control_connection_inactive_time*	Sets internal timer. Defines the timeout for a reply on the control connection. Default: 300 seconds.
LRECL *logical_record_length*	Sends a **SITE LRecl=logical_record_length** command to the local host. Default: None.
MGMTCLASS *SMS_management_class*	Sends a **SITE MGmtclass=SMS_management_class** command to the local host. Default: None.
MIGRATEVOL *migrate_volser*	Default: None.
MYOPENTIME *data_connection_open_time*	Sets internal timer; sends a **SITE DOpen=data_connection_open_time** command to the local host. Defines the timeout when opening a new control or data connection. Default: 300 seconds
NCP *number_of_channel_programs*	Sends a **SITE NCP=number_of_channel_programs** command to the local host. Default: None
PORT local_port	The local host port number used for the control connection. Default: 21
PRIMARY *primary_space*	Sends a **SITE PRImary=primary_space** command to the local host. Default: None.
RDW TRUE | FALSE	If TRUE, the client will send **SITE RDW** to local host; if FALSE, the client will send **SITE NORDW**. Default: None.
RECFM *record_format*	Sends a **SITE RECfm=record_format** command to the local host. Default: None.
RETPD *retention_period*	Sends a **SITE RETpd=retention_period** command to the local host. Default: None.
SECONDARY *secondary_space*	Sends a **SITE SECONDary=secondary_space** command to the local host. Default: None.
SPACETYPE BLOCKS | CYLINDERS | TRACKS	Sends a **SITE BLocks/CYlinders/TRacks** command to the local host. Default: None.

STORCLASS *SMS_storage_class*	Send **SITE STORclass=SMS_storage_class** command to the local host. Default: 300 seconds.
UNITNAME *unit_name*	Sends a **SITE Unit=unit_name** command to the local host. Default: None.
VOLUME *volume_serial*	Sends a **SITE VOLume=***volume_serial* command to the local host. Default: None.
WRAPRECORD TRUE | FALSE	If TRUE, the client will send **SITE WRAPrecord** to local host; if FALSE, the client will send **SITE NOWRAPrecord**. Default: None.

General Client FTP3 Operation

These steps outline the general procedure for using Client FTP3 program:

- Issue the FTP3 command (with any optional parameters) to log on to the remote host. You are automatically logged on to the local MVS TCPaccess host.
- If you do not specify a remote host on your FTP3 command line, you will be prompted for the remote host.
- If the remote host name is in the NETRC file, the user ID and password are taken from the NETRC file. If the host name is not in the NETRC file, you will be prompted for a user ID and password.

  If your user ID and password fail the logon because one or both were entered incorrectly, you must enter the **LOG** command to sign on to the remote host.
- Set the appropriate file transfer parameters (such as MODE, STRUCT, or TYPE).
- Perform the desired transfer operation (such as **GET** and **PUT**).

Path Name

A path name is a string that identifies a file to a file system. A path name must contain a device and/or directory name and a file name. The FTP specification does not specify a standard path name convention. Each user must follow the file naming conventions of the file systems involved in the transfer. Consult the System Administrators at the host sites involved in the transfer for file naming conventions.

Many of the Client FTP3 commands take one or more path name arguments.

For information about the syntax for MVS path names supported by the TCPaccess Server FTP, read “Data Set Names” on page 5-28 in “Server FTP”.

Client FTP3 Command Conventions

"Table 3-3" on page 3-18 provides general notes that apply to the Client FTP3 commands:

Table 3-3 Client FTP3 Command Conventions

Convention	Description
Program Prompt	To indicate successful completion of most commands, the Client FTP3 program gives a new prompt.
Completion of Data Transfer	Final completion of the data transfer command is indicated with a message.
Testing the Control Connections	You can use the **VERBOSE** command to see the specific FTP commands and responses sent and received over the control connections as a result of Client FTP3 commands. If you want more information about this test, refer to **RFC 959**.
Case Sensitivity	The Client FTP3 commands are not case sensitive.
Abbreviations in Commands	Abbreviations are permitted if they are not ambiguous. For example, you can type **AB** for **ABORT** but you cannot type only **A** because several commands begin with this letter.
Brackets in Commands	In the examples, parameters enclosed by brackets are optional for the command line. In many cases, if the optional parameters are omitted from the command line, you are prompted for them.
Syntax Conventions	Command words are shown in uppercase and parameters are shown in lowercase. When the actual values for a parameter are given (such as LSOUTPUT**)**, they are shown in uppercase.
Example Conventions	In all examples of Client FTP3 input and output in this manual, user entries are shown in boldface type. The Client FTP3 examples in this manual assume that you have issued **OPEN** and implied **LOG** commands similar to this example to connect an IBM MVS TSO user to another system:

The NETRC File

The Client FTP3 program uses information in the NETRC file for automatic login to the remote host. The NETRC filename is assumed to be *userid*.NETRC. To use a different naming convention for this file, you must pre-allocate it with an NETRC DD statement.

You can specify any number of MACHINE/LOGIN/PASSWORD sets to define remote hosts. If you define a machine with no login, password, and/or account, you are prompted for this information when needed. When a user connects to a remote host, if a remote MACHINE/LOGIN/PASSWORD set exists for the host, this set logs the user on to the remote host.

If you have not created a NETRC file, you are prompted to supply a user ID and password for the remote host whenever you open a connection to a remote host.

This is the format of the NETRC file; edit this file to specify user IDs, passwords, and accounts, or to create a macro.

`MACHINE` *remhost* `LOGIN` *userid* `(for` *remhost*`) PASSWORD` *password* `(for` *remhost*`)`
`[ACCOUNT` *account* `(for` *remhost*`)`

If an installation does not require a password and/or an account, these can be omitted. Before using the Client FTP3 commands, you must log on to a remote host with an **OPEN** command.

CAUTION ! You should use your local access control facility to protect NETRC files since these files can contain valid user ID password combinations for remote hosts. Only the TSO user ID using the NETRC file should be able to read their own NETRC file.

The NETRC file can also be used in batch jobs by using the NETRC invocation along with a NETRC DD pointing to your NETRC data set.

Note: The batch invocation for NETRC does not assume a default FTP.NETRC data set name.

Transferring and Using a File in a Single JCL Job

Transferring a file in one job step and using that file in another job step can run into a file allocation problem, wherein the file transfer fails in the FTP2 job step. This is usually because the file has been allocated to the batch job and the TCPaccess base product cannot allocate the file for a data transfer. The FTP2 and FTP client programs do not perform file transfers in their own address space; the FTP2 and FTP clients direct the TCPaccess base product to perform file transfers.

As a workaround, perform an IDCAMS ALTER NEWNAME of the file between the file transfer step and the use of the file.

Examples

The following sample JCL does a file transfer, performs an IDCAMS ALTER NEWNAME, and uses the file in the final job step (the data set MVS.P25206.DATA is created in the first job step, renamed to the file MVS.NEWNAME.DATA in the second step, and the file name MVS.NEWNAME.DATA is used in the last step):

```
//MVSA JOB (TSO00...99),'FTP2 BATCH',MSGCLASS=X,NOTIFY=MVS,CLASS=A
//*
//*  JOB TO TRANSFER A FILE AND THEN USE IT IN A LATER JOBSTEP
//*
//*  STEP TO DO THE FTP TRANSFER
//*
//FTP1        EXEC PGM=IKJEFT01,REGION=4000K
//SYSTSIN     DD *,DCB=BLKSIZE=80
    FTP2 / APP=ACCES  TEST NETRC
//STEPLIB     DD DCB=BLKSIZE=32000,
//               DISP=SHR,DSN=T01TCP.LINK
//SYSTSPRT    DD SYSOUT=X
//SYSPRINT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPUT      DD SYSOUT=*,DCB=BLKSIZE=133
//SYSVLT      DD SYSOUT=*,DCB=BLKSIZE=133
//NETRC       DD DISP=SHR,DSN=MVS.FTP.NETRC
//SYSGET      DD *,DCB=BLKSIZE=80
open unix
get temp 'mvs.p25206.data'
END
/*
//*
//*   STEP TO PERFORM AN IDCAMS ALTER NEWNAME
//*
//STEP2       EXEC PGM=IDCAMS
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
 ALTER 'MVS.P25206.DATA'                NEWNAME('MVS.NEWNAME.DATA') -
  CAT(CATALOG.TSO.VESA001)
//*
//*   THIS STEP PRINTS THE FILE ON THE SYSUT1 DD CARD
//*
//STEP3       EXEC PGM=IEBGENER
//SYSUT1      DD DISP=SHR,DSN=MVS.NEWNAME.DATA
//SYSUT2      DD SYSOUT=X,COPIES=1,DCB=*.SYSUT1
//SYSPRINT    DD   SYSOUT=*
//SYSIN       DD   DUMMY
```

The following sample JCL uses a file, performs an IDCAMS ALTER NEWNAME, and does a file transfer in the final job step (the file MVS.OLDNAME.DATA is used in the first jobstep, renamed to the file MVS.TRANSFER.DATA in the second jobstep, and the file MVS.TRANSFER.DATA is transferred in the last jobstep):

```
//MVSB JOB (TSO00,,,99),'FTP2 BATCH',MSGCLASS=X,NOTIFY=MVS,CLASS=A
//*
//* JOB TO TRANSFER A FILE AND USE IT IN A LATER JOBSTEP
```

```
//*
//*   THIS STEP PRINTS THE FILE ON THE SYSUT1 DD CARD
//*
//STEP1     EXEC PGM=IEBGENER
//SYSUT1    DD DISP=SHR,DSN=MVS.OLDNAME.DATA
//SYSUT2    DD SYSOUT=X,COPIES=1,DCB=*.SYSUT1
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    DUMMY
//*
//*   STEP TO PERFORM AN IDCAMS ALTER NEWNAME
//*
//STEP2     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 ALTER 'MVS.OLDNAME.DATA'                NEWNAME('MVS.TRANSFER.DATA') -
  CAT(CATALOG.TSO.VESA001)
//*
//*  STEP TO DO THE FTP TRANSFER
//*
//FTP3      EXEC PGM=IKJEFT01,REGION=4000K
//SYSTSIN   DD *,DCB=BLKSIZE=80
FTP2 / APP=ACCES  TEST NETRC
//STEPLIB   DD DCB=BLKSIZE=32000,
//          DISP=SHR,DSN=T01TCP.LOAD
//SYSTSPRT  DD SYSOUT=X
//SYSPRINT  DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPUT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSVLT    DD SYSOUT=*,DCB=BLKSIZE=133
//NETRC     DD DISP=SHR,DSN=MVS.FTP.NETRC
//SYSGET    DD *,DCB=BLKSIZE=80
open unix
put 'mvs.transfer.data' temp
END
/*
```

Client FTP3 Commands

"Table 3-4" on page 3-22 gives a brief description of each command and its function; detailed descriptions of each command follow the table.

Table 3-4 Client FTP Commands

Command	Function
?	Get help on all commands or one command
ACCOUNT	Send account information
APPEND	Append data to file on remote host
ASCII	Set the transfer type to ACSII
BINARY	Set the file transfer mode to binary
CD	Change working directory on the remote
CDUP	Change to parent directory on remote
CLOSE	Disconnect from remote host
CWD	Change working directory (same as **CD**)
DEBUG	Toggle to activate or deactivate the **DEBUG** option
DELETE	Delete file on remote host
DELIMIT	Display file delimiter on local host
DIR	List directory on remote host
EBCDIC	Set file transfer type to EBCDIC
GET	Copy file from remote host
HELP	Display help information
LCD	Change working directory on local host
LMKDIR	Create a PDS on local host
LOCSITE	Send **SITE** to local host
LOCSTAT	Display status for local host
LPWD	Print the local directory name
LS	List file names on remote host
MDELETE	Delete multiple files on remote host
MGET	Retrieve multiple files from remote host
MKDIR	Create a new directory on remote host
MODE	Set transmission mode
MPUT	Store multiple files on remote
NOOP	Send **NOOP** to remote
OPEN	Open connection to remote host
PASS	Send password to remote host

Command	Function
PROMPT	Toggle prompting for **MGET**, **MPUT** and **MDELETE**
PUT	Store file on remote host
PWD	Show name of working directory on remote host
QUIT	Exit FTP
QUOTE	Send uninterpreted string to remote host
RENAME	Rename file on remote host
RESTART	Restart a transfer
RMDIR	Remove directory on remote host
RUNIQUE	Toggle **STORAGE** method on local host
SENDSITE	Toggle sending of **SITE** command
SITE	Send SITE parameters to remote host
STATUS	Send **STATUS** to remote
STRUCT	Set file structure
SUNIQUE	Store unique file names on remote host
SYSTEM	Display remote host operating system
TSO	Execute **TSO** command
TYPE	Set transfer type
USER	Send userid to remote host
VERBOSE	Display system replies with additional information

? Command

Use the **?** command to display information about using the Client FTP3 program. This command is very similar to the **HELP** command.

? [*command_name* **]**

Syntax Description

command_name	Command for which you are requesting information.

Default

If the **?** command is specified with no arguments, it displays a list of Client FTP3 commands.

Usage Guidelines

- The ? command can be used either with or without arguments.
- If *command_name* is specified as an argument to the **?** command, a line of information displays similar to the information in the table in "Client FTP3 Commands" on page 3-22. The line shows the command syntax and a short description of the command function.

Example

```
? DIR
DIR pathr pathl - Directory list of remote host
```

ACCOUNT

The **ACCOUNT** command displays information that is needed by the remote host. Refer to documentation for the remote FTP server for information that it needs.

ACCOUNT [*account_info* **]**

Syntax Description

account_info	Information to be sent to the remote host.

APPEND

The **APPEND** command requests that a file from the local host be appended to a file at the remote host.

APPEND [*local_path* **] [** *remote_path* **]**

Syntax Description

local_path	Name of the file to be retrieved from the local host.
remote_path	Remote file to which the local file is to be appended.

Default

If either file name is omitted, you are prompted for the file name.

Usage Guidelines

- The syntax for each path depends on the associated Server FTP.
- Data set attributes are maintained for the transferred data set.
- If the data set already exists, and the LRECL is less than that of the transferred data set, the transmitted data set will be truncated.

ASCII

The **ASCII** command sets the data type to ASCII for the data transfer. This command is equivalent to a **TYPE** command with the ASCII (A) parameter specified.

ASCII

BINARY

The **BINARY** command sets the data type to binary for the data transfer.This command is equivalent to a **type** command with the image (I) parameter specified.

BINARY

This command has no arguments or keywords.

CD

The **CD** command requests that the remote Server FTP change the current directory to a new directory.

CD [*path_name* **]**

Syntax Description

path_name	Indicates to the remote Server FTP the name of the directory to be made the current directory.

Default

If you omit *path_name*, the Client FTP3 program prompts you for the desired value.

Usage Guidelines

- The syntax for *path_name* depends on the associated Server FTP.
- If you want to specify a path name that is not a subdirectory of the current directory, enclose the path name in single quotes. Do not leave a space after the first quote; FTP3 will ignore the quotation mark.
- A UNIX Server FTP in session with the Client FTP3 program is using /u/user1/work as the current directory. If a **CD junk** command is issued by the Client FTP3 to that UNIX Server FTP, the resulting current directory is /u/user1/work/junk. The same result is achieved by specifying CD /u/user1/work/junk.

Example

The following example shows a change of directory to a remote UNIX system:

```
CD /u/lpn/d.ddn
250 CWD command successful.
```

The following example shows a change to a directory on a remote system running TCPaccess:

```
CD ACCES
250 "'USER1.ACCES.'" is current prefix
```

The following example shows a change to a specific directory:

```
CD 'TEST.DIR'
250 "'TEST.DIR'" is current prefix
```

CDUP

The **CDUP** command directs the remote Server FTP to change the current directory to the parent directory of the old current directory. The **CDUP** command is most useful when the Server FTP manipulates a hierarchical file system such as UNIX.

CDUP

A UNIX Server FTP in session with the Client FTP3 program has /u/user1/work as the current directory. If a **CDUP** command is issued by the Client FTP3 to that UNIX Server FTP, the resulting current directory is the parent of the old current directory (/u/user1).

Example

The following example shows a change to the parent directory of the old current directory on a remote UNIX system:

```
CDUP
250 CWD command successful.
```

CLOSE

The **CLOSE** command logs you out and terminates the connection between you and the remote Server FTP.

CLOSE

Example

```
CLOSE
221 Goodbye.
```

DEBUG

The **DEBUG** command displays all commands and responses going to and from the Server FTPs and the user. This command is equivalent to specifying the TRACE option on the Client FTP3 command line.

DEBUG

Usage Guidelines

The **DEBUG** command toggles the previous state. If **DEBUG** or **TRACE** was turned on previously, then a subsequent **DEBUG** or **TRACE** command will turn it off.

Example

```
DEBUG
```

DELETE

The **DELETE** command directs the remote Server FTP to delete the specified file.

DELETE [*path_name* **]**

Syntax Description

path_name	The specific file to delete.

Default

If you omit *path_name*, you are prompted to supply one.

Usage Guidelines

The syntax for *path_name* depends on the associated Server FTP.

Example

```
DELETE oldfile
250 DELE command successful.
```

DELIMIT

The **DELIMIT** command displays the character delimiter used between the file name and the file type.

DELIMIT

The delimiter cannot be changed by this command.

DIR

The **DIR** command requests that the remote Server FTP provide a directory list for the specified path.

DIR [*remote_path* **] [***local_path* **] [(DISK]**

Syntax Description

remote_path	Remote directory to be listed. If *remote_path* is not specified, you receive a list of your current directory.
local_path	File to which the directory list is written. If *local_path* is not specified, the directory list is displayed on your screen.
(DISK	Stores output in a data set named *local_directory*.FTP.DIROUTP. If *local_directory* is a PDS, output is stored in member DIROUTP.

Example

The following example lists the contents of a directory on an MVS system.

```
dir /export/home/user1/mvs
150 ASCII data connection for /bin/ls (138.42.224.15,4126) (0 bytes).
-rw-------         1 user1      dvlp       2730 Oct 22        08:36 channel
-rw-------         1 user1      dvlp       2901 Sep  1        09:17 comten
-rw-------         1 user1      dvlp       1276 Oct 21        14:43 dump
-rw-------         1 user1      dvlp        729 Nov 12        05:24 ibmlink
-rw-------         1 user1      dvlp        751 Nov 12        07:41 ibmlink2
226 ASCII Transfer complete.
```

EBCDIC

The **EBCDIC** command sets the data type to EBCDIC for data transfer. The command is equivalent to the **type** command with the EBCDIC (**E**) parameter specified.

EBCDIC

Example

```
EBCDIC
EBCD ENTERED
stat
```

GET

The **GET** command requests that a file from the remote host be copied to a file on the local host by the appropriate Server FTPs. The file to be retrieved is always at the remote host, and the file to be copied into is always at the local host.

GET [*remote_path* **] [** *local_path* **] [(REPLACE]**

Syntax Description

remote_path	File name to be retrieved from the remote host.
local_path	File name at the local host into which the file from the remote host is copied
REPLACE	Specifies that the data set on the local host be overwritten.

Default

If you omit either file name, you are prompted for one.

Usage Guidelines

The syntax for each path depends on the associated Server FTP.

Note: The **FTP GET GDGBASE** command gets only the most recent data set.

The action of the REPLACE option is dependent on the configuration of the FTP statement in your APPCFGxx file.

If OVERWRITE is configured, the file name will be overwritten even without the REPLACE option.

If NOOVERWRITE is configured, the file will not be overwritten even if the REPLACE option is specified.

If SITEOVERWRITE is configured, you must issue a **SITE OVERWRITE** command and specify REPLACE to overwrite an existing file.

HELP

The **HELP** command requests help information from the local Server FTP.

HELP [ALL | ? | *command* **| [SERVER [** *remote_command* **]]]**

Syntax Description

HELP	(Without any parameters) requests a list of commands available to the FTP3 client.
ALL	Provides a brief description of all the commands available to the FTP3 client.
?	Lists the syntax for the **HELP** command.
command	Provides a description of a specific command from the local FTP3 client.
SERVER *remote_command*	Requests information from the remote FTP server. If a specific *remote_command* is specified, a description of the command is returned from the remote server.

Default

If no command is specified, a list of supported commands is returned.

LCD

The **LCD** command changes the current working directory on the local host.

LCD [*local_path_name* **]**

Syntax Description

local_path_name	Specifies the name of the directory on the local host. If you want to specify a path name that is not a subdirectory of the current directory, enclose the path name in single quotes. Do not leave a space after the first quote; FTP3 will ignore the quotation mark.

LMKDIR

The **LMKDIR** command creates a directory (or PDS) on the local host.

LMKDIR [*local_path_name* **]**

Syntax Description

local_path_name	Specifies the name of the directory or PDS that is to be created on the local host. If you want to specify a path name that is not to be appended to the current directory, enclose the path name in single quotes. Do not leave a space after the first quote; FTP3 will ignore the quotation mark.

LOCSITE

The **LOCSITE** command allows you to specify SITE information to be used by the local host.

LOCSITE [*options* **]**

Syntax Description

options	Local site information. For a list of the SITE commands, read the "SITE" on page 5-15.

LOCSTAT

The **LOCSTAT** command displays local FTP status information.

LOCSTAT

LPWD

The **LPWD** command displays the name of the current working directory on the local host.

LPWD

LS

The **LS** command requests a remote Server FTP to provide a list of file names for the specified path.

LS [*remote_path* **] [** *local_path* **] [(DISK]**

Syntax Description

remote_path	Path to be listed from the remote host.
local_path	Local file into which the list from the remote server is printed.
(DISK	Stores output in a data set named *local_directory*.FTP.LSOUTPUT. If *local_directory* is a PDS, output is stored in member LSOUTPUT.

Default

If a local path is not specified, the list of files appears on your screen.

Usage Guidelines

- If the path specifies a directory or other group of files, the remote Server FTP transfers a list of files.
- The syntax for each path depends on the associated FTP server.
- If a local path is specified, the list of files is written to the specified file.

Example

The following example shows of the **LS** command with parameters:

```
ls /export/home/user1 unix.dir.temp
150 ASCII data connection for /bin/ls (138.42.224.15,4134) (0 bytes).
226 ASCII Transfer complete.
 350 bytes received in 1.54 seconds (227 bytes/s)
```

This is an example of the **LS** command without parameters:

```
ls
150 ASCII data connection for /bin/ls (138.42.224.15,4135) (0 bytes).
-Transfer complete
channel
comten
dump
filea
tempfile
226 ASCII Transfer complete.
```

MDELETE

The **MDELETE** command allows you to delete multiple files on the remote host.

MDELETE [*remote_path_name* **] | [** *mask* **]**

Syntax Description

remote_path_name	Name of the file to be deleted on the remote host.
mask	Mask to use to describe files. This mask complies with mask conventions on the remote host.

Example

```
MDELETE userid.myfiles.*
MDELETE file199?
```

MGET

The **MGET** command copies multiple files from the remote host to the local host. You cannot specify a different name for the files that are retrieved. Filenames created on the local host will be identical to those on the remote host.

MGET [*remote_path_name* **] [(REPLACE]**

Syntax Description

remote_path_name	Name of the files to be copied from the remote host.
REPLACE	Specifies that the data set on the local host be overwritten.

Note: The action of the REPLACE option is dependent on the configuration of the FTP statement in your APPCFGxx file.

If OVERWRITE is configured, the file name will be overwritten even without the REPLACE option.

If NOOVERWRITE is configured, the file will not be overwritten even if the REPLACE option is specified.

If SITEOVERWRITE is configured, you must issue a **SITE OVERWRITE** command and specify REPLACE to overwrite an existing file.

Example

```
ftp> lcd mget.pds
"'GAM2.MGET.PDS'" partitioned data set is current directory
ftp> cd tstdir
---> CWD tstdir
250 CWD command successful.
ftp> mget TST*
---> PORT 138,42,160,55,17,34
200 PORT command successful.
---> NLST TST*
150 ASCII data connection for /bin/ls (138.42.160.55,4386) (0 bytes).
226 ASCII Transfer complete.
GET TST1
---> PORT 138,42,160,55,17,35
200 PORT command successful.
550 Catalog structure invalid or user lacks authority to catalog
GET TST2
---> PORT 138,42,160,55,17,36
200 PORT command successful.
---> RETR TST2
150 ASCII data connection for TST2 (138.42.160.55,4388) (29 bytes).
226 ASCII Transfer complete.
GET TST3
---> PORT 138,42,160,55,17,37
200 PORT command successful.
---> RETR TST3
```

```
150 ASCII data connection for TST3 (138.42.160.55,4389) (37 bytes).
226 ASCII Transfer complete.
ftp>
```

MKDIR

The **MKDIR** (make directory) command directs a remote Server FTP to create the specified directory.

MKDIR [*path_name* **]**

Syntax Description

path_name	Directory to be created.

Default

If you omit *path_name*, you are prompted for it.

Usage Guidelines

- If the path name is relative, the specified subdirectory is created in the current working directory.
- If the path name is absolute, the specified directory is created.
- The syntax for path depends on the associated Server FTP.

Example

A UNIX Server FTP in session with the Client FTP3 program has /u/user1/work as the current directory. If a **mkdir junk** command is issued by the Client FTP3 to that UNIX Server FTP, the subdirectory junk is created in the current directory (/u/user1/work/junk). The same result is achieved by specifying **MKDIR /u/user1/work/junk**.

```
MKDIR /u/lpn/d.new
257 MKD command successful.
```

MODE

The **MODE** command sets one of three transmission modes:

MODE S | B | C

- Block mode

  Block mode formats the data and allows for restart procedures.

- Stream mode

  Stream mode passes the data with little or no processing. It interacts with the structure attribute to determine the type of processing. Stream mode is the default if no **MODE** command was used.

- Compressed mode

  Compressed mode transmits the data blocks with no repetitive characters or blanks.

  For the purpose of standardized transfer, the sending host translates its internal end-of-line or end-of-record representation into the representation required by the transfer mode and file structure, and the receiving host performs the inverse translation to its internal representation. Since these transformations make extra work for some systems, identical systems transferring non-record structured text files might use binary representation and stream mode to simplify transfer.

Syntax Description

S	Stream mode.
B	Specifies block mode.
C	Compressed mode. Data transfer type must be set to EBCDIC.

Usage Guidelines

One of the three codes (either S, B, or C) is required as an argument on the **MODE** command.

Each of the possible transmission modes is discussed in the following sections. For a detailed description of the effect of various transmission modes, read the “Transmission Modes” section in **RFC 959**, File Transfer Protocol.

Not all Server FTPs support all transmission modes; review the Server FTP documentation if you have questions concerning transmission mode support.

Block Mode

Block mode is indicated with the character B. In block mode, the file is transmitted as a series of data blocks preceded by one or more header bytes. Record structures are allowed in this mode, and any representation type can be used. Restart markers are embedded in the data stream.

Stream Mode

Stream mode is set with the character **S**. This is the default if no **MODE** command has been used. In stream mode, the data is transmitted as a stream of bytes. There are no restrictions on the representation type used, and record structures are allowed. In a record structured file, End of Record (EOR) and End of File (EOF) are each indicated by a two-byte control code included with the data sent over the data connection. If the structure is a file structure, the EOF is indicated by the sending host closing the data connection, and all bytes sent over the data connection are data bytes.

Compressed Mode

Compressed mode is set with the character C. In compressed mode, data blocks are transmitted with one or more header bytes. Logical record boundaries of the data set are preserved with compressed mode. Data is transmitted with no repetitive characters or blanks.

MPUT

The **MPUT** command copies multiple files from the local host to the remote host.

You cannot specify a different name for the files that are transferred. Filenames created on the remote host will be identical to those on the local host.

MPUT [*local_path_name* **] | [** *mask* **]**

Syntax Description

local_path_name	File name on the local host.
mask	Mask to use to describe files. This mask complies with mask conventions on the remote host.

Example

```
MPUT userid.myfiles.*
MPUT file199?
```

NOOP

The **NOOP** command is used to test the response of the remote host.

NOOP

OPEN

The **OPEN** command is used to open a connection to the FTP server on the remote host. This command can be issued while you are in the FTP environment.

OPEN *remote_host* **[** *port_number* **]**

Syntax Description

remote_host	Name of the remote host. Client FTP3 will automatically connect to this host at initialization. This parameter is required. If you do not supply this parameter, you will be prompted for it.
port_number	Port number of the FTP server on the remote host. Default: 21

Note: If you are currently connected to a remote host, you must disconnect before using the **open** command to connect to another host

PASS

The **PASS** command is used to send a password to the remote host.

PASS *password*

Syntax Description

password	Specifies your password on the remote host.

Usage Guidelines

The PASS command must follow the USER command.

Read “USER” on page 3-71 for more information.

PUT

The **PUT** command requests that the local Server FTP copy a file to the remote system. The file to be copied is always at the local Server and the file destination is always at the remote Server.

PUT [*local_pat* **] [** *remote_path* **]**

Syntax Description

local_path	File name of the file to be copied from the local server.
remote_path	File name of the file at the remote Server into which the file from the local server is copied. If no name is specified, the name from the local host is used.

Usage Guidelines

The syntax for each path depends on the associated Server FTP. If you omit either path, you are prompted for the file name.

PWD

The **PWD** command directs a remote Server FTP to return the path name of the current working directory.

PWD

Syntax Description

This command has no arguments or keywords.

Example

```
PWD
257 "/u/lpn" is current directory.
```

QUIT

The **QUIT** command terminates the Client FTP3 program.

QUIT

Example

```
QUIT
221 Goodbye.
```

QUOTE

The **QUOTE** command sends an uninterpreted, unaltered character string to the remote Server FTP over the control connection. This mechanism sends FTP commands to the Server that the Client FTP3 program might not be able to send.

QUOTE [*text* **]**

Syntax Description

text	Sent to the Server over the control connection exactly as you enter it; if the text is omitted, you are prompted to enter it

Example

```
QUOTE pasv
227 Entering Passive Mode (26,131,0,17,4,216).
```

RENAME

The **RENAME** command directs a remote Server FTP to rename a file.

RENAME [*old_path_name* **] [** *new_path_name* **]**

Syntax Description

old_path_name	File name to be renamed.
new_path_name	New name to be assigned to that file.

Default

If you omit either argument, you are prompted to enter it.

Usage Guidelines

The syntax of the path names depends on the associated Server FTP.

Example

```
RENAME titlecol coltitle
350 File exists, ready for destination name
250 RNTO command successful.
```

RESTART

The **RESTART** command restarts the last checkpointed file transfer.

REST

The data set *USERID*.FTP.CHKPOINT on the local host holds the valid checkpoint for the last checkpointed file transfer. Any parameters that may have changed for the last transfer, such as transfer mode and type, must be re-set before restarting the transfer.

When file transfer is in blocked or compressed mode (mode=b/c), and Client FTP3 processes a **GET** or **PUT** command, it allocates a fixed-block data set named *userid*.FTP.CHKPOINT. It uses this file to save the **GET** or **PUT** command and checkpoint record returned from the FTP server. If the *userid*.FTP.CHKPOINT file already exists, Client FTP3 will rewrite the checkpoint data set from the beginning of the data set.

If the **GET** or **PUT** command is successful, the checkpoint data set is deleted upon completion of the command processing. If the **GET** or **PUT** command fails or is interrupted, the *userid*.FTP.CHKPOINT data set will be kept in the system for the **RESTART** command to use to restart the file transfer.

If the **RESTART** command is executed successfully, Client FTP3 will delete the checkpoint data set. If the **RESTART** command fails or is interrupted, the checkpoint data set is kept for future use.

Users are responsible for deleting the checkpoint data set once it is determined the data set is no longer needed.

For more information, read about the **RESTART** interval option for the **SITE** command in “SITE” on page 5-15.

RMDIR

The **RMDIR** (remove directory) command directs a remote Server FTP to remove the specified directory.

RMDIR [*path_name* **]**

Syntax Description

path_name Directory to be removed.

Default

If you omit *path_name*, you are prompted for it.

Usage Guidelines

- If the path name is relative, the specified subdirectory is removed from the current working directory.
- If the path name is absolute, the specified directory is removed.
- The syntax of *path_name* depends on the associated Server FTP.

Note: Many systems require the directory to be empty before it can be removed.

Example

As an example, if a UNIX Server FTP in session with the Client FTP3 program has /u/user1/work as the current directory, and a **RMDIR junk** command is issued by Client FTP3 to that UNIX Server FTP, the junk subdirectory of the current directory is removed. The same result is achieved by specifying **RMDIR /u/user1/work/junk**.

```
RMDIR /u/lpn/d.samp
250 RMD command successful.
```

SENDSITE

The **SENDSITE** command is used to automatically send the **SITE** commands when sending a file to a remote host. This commands is useful only with other IBM-based FTP servers.

SENDSITE

The **SENDSITE** command is turned on when you start your FTP session. You can toggle it on and off by issuing the **SENDSITE** command. To determine the current state of the **SENDSITE** command, issue the **LOCSTAT** command.

SITE

The **SITE** (site parameters) command provides the local Server FTP with specific information it requires. This information is essential to file transfers involving that Server FTP, but is not sufficiently universal to have been included specifically in the FTP. Typically, you use a **HELP SITE** Client FTP3 command to find the **SITE** requirements for a specific local Server FTP. Otherwise, review the Server FTP documentation for the **SITE** requirements. **SITE** command parameters are described in “SITE” on page 5-15.

SITE *text*

The *text* argument is required and is passed through unchanged to the specified server.

STATUS

The **STATUS** command requests status information from the remote system.

STATUS [*path_name* **]**

Syntax Description

path_name	Directory or file name for which information is requested.

STRUCT

The **STRUCT** (file structure) command provides information on file structure to the remote Server FTP.

STRUCT F | R

Syntax Description

F	File structure is specified by F. This is the default if no **STRUCT** command has been used. File structure is used for files with no internal structure, and the file is considered to be a contiguous sequence of data bytes. File structure is accepted for text files (i.e., files with type ASCII or EBCDIC) by all FTP implementations.
R	Record structure is set by R. This is for files made up of sequential records.

Usage Guidelines

Record structure is set by R. This is for files made up of sequential records.

Example

```
STRUCT  F
```

SUNIQUE

Use the **SUNIQUE (STORE UNIQUE)** command to store transferred files by unique file names on a remote machine. **SUNIQUE** is a toggle command (meaning it is either off or on). When used, the target server automatically ensures that files received in FTP transfer are stored under a unique name.

SUNIQUE

Usage Guidelines

- The target remote FTP server must support the **STOU** (store unique) command.
- If **SUNIQUE** is not used, FTP3 will issue the standard **STOR** command. If file names are not unique, files in the target directory could be overwritten.
- Default for **SUNIQUE** is **OFF**.

SYSTEM

The **SYSTEM** command displays the name of the operating system on the remote host.

SYSTEM

Example

```
SYSTEM
215 MVS is the operating system of this server.
```

TRACE

Use the **TRACE** command to activate or deactivate the tracing option. With the tracing option active, you can rerun failing commands to discover the reason for the failure.

TRACE

Usage Guidelines

The **TRACE** command toggles the previous state. If **TRACE** or **DEBUG** was turned on previously, then a subsequent **TRACE** or **DEBUG** command will turn it off.

Example

```
DEBUG
```

TSO

The **TSO** command requests the client FTP3 program to execute a TSO command for you.

TSO *command*

Syntax Description

command	Specifies the TSO command.

TYPE

The **TYPE** command tells a Server FTP the data type to use.

TYPE I | L *byte_size* **| { A | E [N | T | C] }**

Syntax Description

I	Indicates image type. The data is sent as a contiguous bit stream that, for transfer, is packed into 8-bit transfer bytes. The receiving site stores the data as contiguous bits. The receiving storage system might need to pad the file (or each record, in record-structured files) to some convenient boundary. Review the documentation for a Server FTP to find out about padding. Image type is for the efficient storage and retrieval of files and for transfer of binary data. All FTP implementations are required to support the image type.
L *byte_size*	Indicates the local file type and the logical byte size of the file. The byte size value (*byte_size*), representing the logical byte size, is required with the local type. With this type, the data is transferred in logical bytes of the specified size. The logical byte size might differ from the transfer byte size. If the logical and transfer byte sizes differ, the logical bytes are packed contiguously, disregarding transfer byte boundaries, and are padded at the end if necessary. When the data reaches the receiving host, it is transformed in a manner dependent on the logical byte size and the particular host. The transformation is invertible; an identical file can be retrieved if the same parameters are used.
A	Sets the file type to ASCII. This type is accepted by all FTP implementations and is good for transferring text files, except when both hosts find the EBCDIC type more convenient. In accordance with the *NVT* standard, the CRLF sequence is used at the end of a line of text. The sender converts the data from an internal character representation to the standard 8-bit NVT ASCII representation (see the TELNET specification in the list of reference documents). The receiver converts the data from this standard form to the receiver's own internal form. **Note** **A** is the default argument for the **TYPE** command.

E

Sets the files type to EBCDIC. This type performs efficient transfer between hosts that use EBCDIC. TCPaccess Client FTP3 users usually use this type when copying files to their MVS host.

For transmission, data is 8-bit EBCDIC characters. The character code is the only difference between EBCDIC and ASCII types.

End-of-line is rarely used with EBCDIC type to denote structure, but where it is necessary, the NL character is used.

The types ASCII and EBCDIC optionally take a second parameter that indicates what kind of vertical format control, if any, is associated with a file. If a file is to be sent to a host for printing, the receiving host must know how the vertical format control is represented. Therefore, the ASCII and EBCDIC types have a second parameter specifying non-print, TELNET, or carriage control (ASA).

These are the vertical format control specification options:

blank	Move paper up one line.
0	Move paper up two lines.
-	Move paper up three lines.
1	Move paper to top of next page.
+	No movement (i.e., overprint).

Default

ASCII is the default file type.

For both ASCII and EBCDIC file types, vertical format control **N** is the default.

Usage Guidelines

- One of the four arguments (I, L *byte_size*, A, or E) is required.
- If local type (L) is set, the integer byte size argument must also be set.
- If ASCII (A) or EBCDIC (E) type is set, one of the three vertical format control arguments, N, T, or C, can also be set.

USER

The **USER** command identifies your userid to the remote FTP server.

USER *user_id* **[** *password* **] [** *old_password* **] [/** `new_password` **]**

Syntax Description

user_id	Your login name on the remote host.
password	Your password on the remote host. You will be prompted for your password if you do not supply it.
old_password	Your current password on the remote host.
new_password	Your new password on the remote host.

Usage Guidelines

- You can issue the USER command to change your userid at any time while you are in the FTP environment.
- Your password will not be printed if you allow the client FTP3 to prompt you for it.
- For more information about using the NETRC file, read “The NETRC File” on page 3-19.

4

Client FTP

This chapter describes Client FTP, the File Transfer Protocol (FTP) that supports file transfers among unlike hosts in diverse internetworking environments. It contains these sections:

- “Introducing Client FTP” on page 4-3

  Provides a brief overview of the File Transfer Protocol.

- “Client FTP” on page 4-4

  Illustrates how FTP works.

- “Invoking Client FTP” on page 4-5

  Describes how to use Client FTP as both a TSO command processor and as a regular batch program.

- “Invoking FTP as a Batch Program” on page 4-7

  Describes how to run the Client FTP program in batch as either a regular program or as a TSO command processor.

- “Client FTP Invocation Options” on page 4-9

  Describes both the general and debug options available with the FTP command.

- “General Client FTP Operation” on page 4-12

  Describes the general operation of the Client FTP program.

- “Client FTP Commands” on page 4-14

  Describes each of the Client FTP commands and includes a table listing all the commands with a brief description of each.

  Commands are:

“? Command” on page 4-16	“ABOR” on page 4-17	“ADD” on page 4-18	“ALLO” on page 4-19
“BYE” on page 4-20	“CDUP” on page 4-21	“CONN” on page 4-22	“CWD” on page 4-23
“DELE” on page 4-24	“DO” on page 4-25	“END” on page 4-26	“EXPE” on page 4-27
“GET” on page 4-28	“HELP” on page 4-29	“LIST” on page 4-30	“LOG” on page 4-31

“MKD” on page 4-32	“MODE” on page 4-33	“NLST” on page 4-35	“PUT” on page 4-36
“PWD” on page 4-37	“QUIT” on page 4-38	“QUOT” on page 4-39	“REN” on page 4-40
“REST” on page 4-41	“RMD” on page 4-43	“SEND” on page 4-44	“SITE” on page 4-45
“SNDS” on page 4-46	“STAT” on page 4-47	“STRU” on page 4-48	“TYPE” on page 4-49
“A=B” on page 4-51	“A?B” on page 4-52		

- “Restart Support” on page 4-53

  Describes how to use the restart marker.

- “Client FTP File Transfer Examples” on page 4-54

  Provides examples of some of the Client FTP commands.

Introducing Client FTP

The File Transfer Protocol (FTP) is an application protocol in the internet protocol suite. It supports file transfers among unlike hosts in diverse internetworking environments. Using FTP, you can move a file from one computer to another, even if each computer runs a different operating system and uses a different file storage format. Files can contain data, programs, text, or anything that can be stored online.

The objectives of the FTP protocol are to:

- provide sharing of files (computer programs and/or data)
- encourage indirect or implicit (via programs) use of remote computers
- shield users from variations in file storage systems among hosts
- transfer data reliably and efficiently

The FTP is based on a model of files having a few attributes and a mechanism of commands and replies. The command - reply mechanism establishes the parameters for a file transfer and then performs the transfer. Like TELNET, FTP runs over TCP and assumes the service level provided by TCP.

These documents define FTP:

- *RFC 959, File Transfer Protocol (FTP)*
- *MIL-STD-1780, Military Standard File Transfer Protocol*

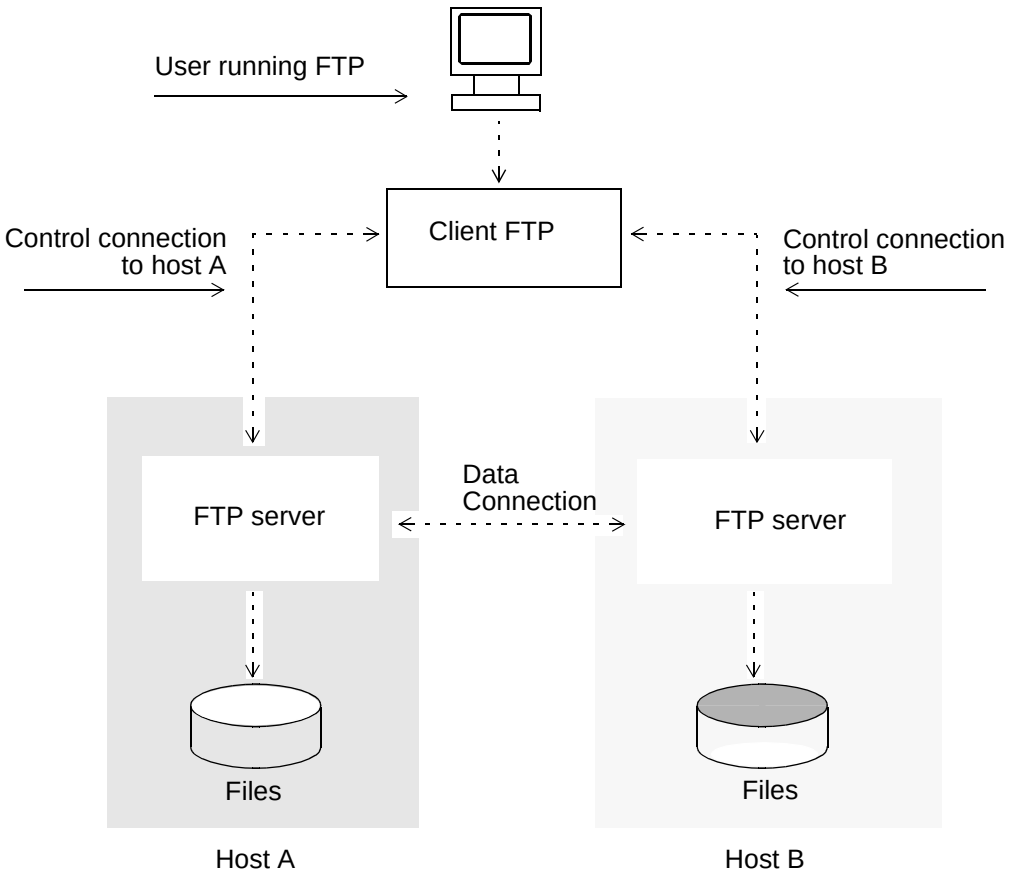
Two versions of Client FTP are provided with TCPaccess: Client FTP and Client FTP2. This chapter describes Client FTP.

The TCPaccess Client FTP program uses a three-party model, or the server-to-server model. Client FTP requires the MVS TSO or batch user to sign on to two hosts that have Server FTP programs. After signing on to the two hosts, a user can transfer files between host **A** and host **B**.

Client FTP

"Figure 4-1" on page 4-4 shows the relationship between the Client FTP program and the two Server FTP programs in the three-party model.

Figure 4-1 Client FTP Program in the Three-Party Model



Through FTP, the user opens connections to two FTP servers. Both servers may be remote, or one may be a local server and the other a remote server. FTP maintains two control connections: One for communication with the TCPaccess FTP server and the second with the remote FTP server. The user issues command sequences to direct both servers.

Client FTP gives the user access to files on both systems. Data transfer occurs directly between the local and remote servers. Data does not pass through the Client FTP application.

Invoking Client FTP

The Client FTP program runs as both a TSO command processor and as a regular batch program. This means that Client FTP can be used as a TSO command and can be called as a regular program with MVS JCL.

TSO Invocation

In a TSO environment, Client FTP can be accessed as a TSO command processor or can be called as a program with the TSO **CALL** command.

Since Client FTP does not use full-screen facilities, it can be used from any type of terminal that is supported by TSO including 3270 systems, 3767 systems, and asynchronous ASCII terminals supported through TCAM, NTO, or NPSI.

Note: You must have **PROMPT** set in your TSO profile for FTP to work properly in interactive mode.

FTP Command Processor

You can invoke Client FTP with the FTP TSO command as shown in the following example:

FTP [*/ option1 option2* **...]**

Client FTP responds with this message:

```
TCPaccess Rn Client FTP - Enter command or '?'
```

Read "Client FTP Invocation Options" on page 4-9 for a description of the options for the Client FTP commands.

Note: All FTP options must be preceded by a slash (/). If this character is omitted, the options are not recognized by Client FTP.

TSO CALL Command

Use the TSO **CALL** command in a TSO environment to invoke Client FTP.

CALL 'T01TCP.LINK(FTP)' [*'/option1 option2 ..'*]

Note: The data set name, **T01TCP.LINK**, might need to be replaced by the appropriate data set name at your installation. Check with your TCPaccess site administrator.

Usage Guidelines

- Options, when specified, must be enclosed in single quotes.
- When invoked by **CALL**, Client FTP runs as a program and not as a TSO command processor. No data sets need be allocated before invoking FTP in this fashion.
- Under TSO, any data set needed by the Client FTP program is dynamically allocated and freed.

Invoking FTP as a Batch Program

The Client FTP program can be run in batch as either a regular program or as a TSO command processor by running it under a batch Terminal Monitor Program (TMP).

Note: When running in batch mode, it is important to specify the Client FTP commands carefully, because the slightest error can cause all subsequent commands to fail and force you to rerun the batch job. For this reason, you should execute only one file transfer per batch job step.

For PL/I V2.2.1 users, FMID PLIX150 must be installed. The TCPaccess **LINK** library must be placed before PL/I runtime libraries in your **STEPLIB** concatenation for client batch jobs (including **FTP**, **FTP2**, **ACPEEP**, or **TELNET**). Failure to do this may cause IBM002I, IBM004I, or IBM014I messages.

When run in batch, Client FTP sets a program condition code depending on the severity of Client FTP errors encountered; a return code of 0 indicates all commands entered processed successfully.

Batch Program

You can invoke Client FTP in batch in a manner similar to any other batch utility program.

Note: The **FIOS** option is required in the batch environment.

Example

The following example downloads the host names table from the Network Information Center (NIC) to an existing sequential data set on your host.

```
//jobname   JOB job_stmt_parms
//FTP EXEC PGM=FTP,PARM='/FIOS option1 option2 ..'
//STEPLIB   DD DSN=T01TCP.LINK, DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSPUT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSGET    DD *,DCB=BLKSIZE=80
CONN NIC
CONN your_hostname
A:LOG ANONYMOUS GUEST
B:LOG your_userid your_password
B:SITE BLKSIZE(4000) LRECL(260) RECFM(VB) SPACE(xx,y)
PUT <NETINFO>HOSTS.TXT 'your_hoststxt_dsname'
A:QUIT
B:QUIT
END
/*
```

Batch TMP

You can invoke Client FTP in batch as a TSO command processor by running it under a batch TMP.

Example

In the following example, the batch TMP program is **IKJEFT01**, which is the normal TSO TMP. The following example downloads a Request For Comments (RFC) from the NIC into a member of an RFC partitioned data set on your host:

```
//jobname  JOB job_stmt_parms
//TMP EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*,DCB=BLKSIZE=133
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
FTP / FIOS
//*
//SYSPUT DD SYSOUT=*,DCB=BLKSIZE=133
//SYSGET DD *,DCB=BLKSIZE=80
CONN NIC
CONN your_hostname
A:LOG ANONYMOUS GUEST
B:LOG your_userid your_password
TYPE AT
PUT <RFC>RFCxxx.TXT 'your_rfctxt.dsname(RFCxxx)'
A:QUIT
B:QUIT
END
/*
```

Client FTP Invocation Options

This section describes the Client FTP invocation options and their requirements. These general notes apply to the Client FTP invocation options:

- No invocation options are required for the Client FTP program. If any are specified, they must be preceded by a slash (/) to meet the conventions of the PL/I runtime support package. In the previous examples a blank follows the (/), but it is not required.
- An option name can immediately follow a slash, as in **/FIOS**.
- Invocation options are not case sensitive.
- Specify option names exactly as shown. Abbreviations are not permitted.

Client FTP uses general options and debugging options. They are described in the following sections.

Client FTP General Options

This section describes the Client FTP general options.

APP

The **APP** option (in the form **APP=***vtam_application_name*) identifies the exact VTAM application name where the client will connect. *vtam_application_name* is a 1- to 8-character name.

If the **APP** option is used, the **SYS** invocation option is ignored.

FIOS

Normally the FTP program interacts with the user through a terminal. The **FIOS** option lets the program read and execute a file containing commands and sends the results to a different file. Commands are read from a sequential file allocated to the **SYSGET DD** statement; execution results are written to a sequential file allocated to the **SYSPUT DD** statement.

Example

Under TSO, the files can be allocated as shown here:

```
ALLOCATE FILE(SYSGET) DATASET(input_dataset)
ALLOCATE FILE(SYSPUT) DATASET(output_dataset)
```

The input file should have all the necessary commands (such as, **CONN**, **LOG**) and should be unnumbered. The output file should have a blocksize of 133.

Note: You must use the **FIOS** option when running FTP in batch.

LOGT

The **LOGT** option displays the current time before each line is sent to the terminal. This option is automatically set if either the **TEST** or **FIOS** option is specified.

SYS

The **SYS** option, in the form **SYS=***x*, where *x* is an arbitrary character, identifies an alternate TCPaccess VTAM application to handle the TELNET connections established by the Client FTP program. This option is useful in cases where multiple copies of TCPaccess are running concurrently. You can specify **SYS=***x* to access a network server called **ACCES***x* instead of the usual **ACCES**.

WAIT

The **WAIT** option causes the terminal keyboard to remain locked while a data transfer is in progress. Normally the keyboard remains unlocked, allowing additional commands to be entered during a data transfer.

This option is automatically set if the **FIOS** option is specified.

Client FTP Debugging Options

The Client FTP debugging options obtain debugging information on the internal operation of the Client FTP program and the interactions between the Client FTP program and the Server FTPs. Normally, the debugging information displays on the terminal, but if the **FIOS** option is in effect, the information is written to the data set represented by the **SYSPUT DD** statement.

Note: Use these options only under the direction of TCPaccess Customer Support.

DISP

Use the **DISP** option to display all requests sent and all responses received on the control connections to each Server FTP.

Examples

Requests sent from the Client FTP program to a Server FTP use the following format:

DISP - OUTPUT - *text of User-to-Server request*

Responses sent from a Server FTP to the Client FTP program have the following format:

DISP - *text of Server-to-User response*

TEST

Use the **TEST** option to obtain detailed status information from the various internal calls in the Client FTP program as well as the information provided by the **DISP** option. This option can be turned on or off while the program is running by issuing the **DEBUG** command.

Example

Information logged by the **TEST** option has the following format:

TEST - *TEST debugging information text*

TESTI

Use the **TESTI** option to obtain local terminal input and output information, as well as to provide the **TEST** and **DISP** information.

Example

Information logged by the **TESTI** option has the following format:

TESTI - *TESTI debugging information text*

VLT

The **VLT** option turns on tracing of the virtual line terminal sessions associated with the FTP session. This option is useful for debugging VTAM problems between the FTP session and the TCPaccess address space. The **VLT** option generates an enormous amount of output. When used interactively, this output comes to the terminal. When used in batch, the output is written to the **SYSVLT DD**. Thus, when running FTP in batch, you must add this **DD** statement to the JCL:

Example

```
//SYSVLT   DD    SYSOUT=*,DCB=BLKSIZE=133
```

General Client FTP Operation

These steps outline the general procedure for using this Client FTP program:

- Connect to the two host Server FTPs using the **CONN** *host* command. The first host connected is the A: host, and the second is the B: host.
- Use these commands to log in to each host Server FTP:

A:LOG *userid_a password_a*

B:LOG *userid_b password_b*

- Set the appropriate file transfer parameters (such as **MODE**, **STRU**, **TYPE**).
- Perform the desired transfer operation (such as **GET**, **PUT**).

Path Name

A path name is a string that is provided to a file system to identify a file. A path name usually contains a device and/or directory name and a file name. The FTP specification does not specify a standard path name convention. You must follow the file naming conventions of the file systems involved in the transfer. Consult personnel at the host sites involved in the transfer for file naming conventions.

Many of the Client FTP commands take one or more path name arguments.

For information about the syntax for MVS path names supported by the TCPaccess Server FTP, read "Data Set Names" on page 5-28 in "Server FTP".

Client FTP Command Conventions

These general notes apply to the Client FTP commands:

- To indicate successful completion of most commands, the Client FTP program gives a new prompt. However, when a data transfer command is issued, a prompt appears when the operation begins successfully. You can then enter other commands (such as, status requests with the **STAT** command) while the operation proceeds. Final completion of the data transfer command is indicated with a message.
- You can use the **DISP** invocation option to see the specific FTP commands and responses (according to ***RFC 959***) sent and received over the control connections as a result of Client FTP commands.
- The Client FTP commands are not case sensitive.
- The commands must be specified exactly as shown. Abbreviations are not permitted.
- In the examples, parameters enclosed by brackets are optional for the command line. In many cases, if the optional parameters are omitted from the command line, you are prompted for them.
- In all examples of Client FTP input and output in this manual, user entries are shown in boldface type.
- The Client FTP examples in this manual assume that you have issued **CONN** and **LOG** commands similar to the following example to connect an IBM MVS system to another system:

Example

```
TCPaccess Rn Client FTP - Enter command or '?'
conn mvs
A:220 MVS.HQ.COMPANY.COM -- FTP Server, Enter command or HELP
conn unix
B:220 unix FTP server (SunOS 4.1) ready.
a:log myid
331 Enter PASS command
Password:
b:log myid
331 Enter PASS command
Password:
```

The Host Prefix

Because the Client FTP program implements the three-party model in which the Client FTP program establishes control connections to two Server FTPs, you need a way to identify which connection is to be used for certain commands. For example, you might want to provide one set of site parameters with the **SITE** command to one Server FTP and provide a different set of parameters to the other Server FTP. This Client FTP program uses host prefixes to differentiate between the two Server FTPs.

A host prefix is one of the A: or B: character strings. When specified with a Client FTP command, the host prefix precedes the command, as in **A:SITE.** A host Server FTP initially is assigned to one prefix or the other when the connection is established with the **CONN** command. The **A=B** command can switch the host prefixes from A: to B: and B: to A: when necessary. The **A?B** command shows the host name associated with each host prefix in case you become confused.

Client FTP command host prefixes can be required, not allowed, ignored, or optional.

- Commands that require a host prefix must be preceded by an A: or B:.
- Commands for which a host prefix is not allowed must not be preceded by an A: or a B:.
- Commands for which a host prefix is ignored can either have one or not; in any case, the prefix is ignored.
- Commands for which a host prefix is optional can either have a prefix, or not. If you give a prefix, the command applies only to that specific host. If you omit the prefix, the command applies to both hosts, provided the connection to the host is open. The exception is the **CONN** command. If a prefix is omitted, the first available, unused host prefix is assigned to the connection.

Client FTP Commands

This section describes the Client FTP commands. “Table 4-1” on page 4-14 gives a brief description of each command and its function. The remainder of this chapter provides detailed descriptions of the commands.

Table 4-1 Client FTP Commands

Command	Function	Host Prefix Requirement
?	Get help on all commands or one command	Ignored
ABOR	Abort transfer data	Optional
ADD	Append a file from host A to a file at host B	Not allowed
ALLO	Set file size	Required
BYE	Terminate program (same as **QUIT**)	Required
CDUP	Change to parent of current working directory	Required
CONN	Control connection between Client FTP program and a Server FTP	Optional
CWD	Change current directory	Required
DELE	Delete a file on the remote host	Required
DO	Execute TSO command	Ignored
END	Terminate program	Not allowed
EXPE	Toggle experimental mode for directory commands	Ignored
GET	Copy file from remote host to local host (same as **RECV**)	Not allowed
HELP	Ask local host for command information	Optional
LIST	Provide current information for files in a specified path name	Not allowed
LOG	Login user	Required
MKD	Make a directory	Required
MODE	Set transmission mode	Optional
NLST	List filenames in specified directory	Not allowed
PUT	Copy file from local host to remote host (same as **SEND**)	Not allowed
PWD	Show name of current working directory (on the remote host)	Required
QUIT	Terminate program (same as **BYE**)	Optional
QUOT	Send an FTP command to the remote Server	Required
REN	Rename file from old name to new name (on the remote host)	Required
REST	Restart	Required
RMD	Send an FTP command to the remote Server	Required
SEND	Copy file from local host to remote host (same as **PUT**)	Not allowed
SITE	Send local host-dependent information	Required
SNDS	Re-send last **SITE** command to local host	Required

Command	Function	Host Prefix Requirement
STAT	Ask local host for status	Optional
STRU	Set file structure	Optional
TYPE	Set data type	Optional
A=B	Switch host prefixes associated with each Server FTP connection	Ignored
A?B	Show current connections associated with the A and B host prefixes from the Client FTP program	Ignored

? Command

The **?** command obtains information about using the Client FTP program.

? [*command_name*]

Syntax Description

command_name	Command for which information is requested.

Default

If no arguments are specified, a list of Client FTP commands is displayed.

Usage Guidelines

- Each command name is preceded by a special character that indicates the requirement for host prefixing for that particular command. A key is provided in the display to explain the prefix requirement indicated by each special character.
- If a command name is specified as an argument on the **?** command, a line of information appears, showing
  - the command syntax
  - a short description of the command function
  - host prefix requirements for that command
- The **?** command ignores a host prefix.

Example

The following example shows a request for information about using the **log** command.

```
? log
LOG <USERID> <CURRENT_PASSWORD> </NEW_PASSWORD> - LOGIN USER
(PREFIX REQUIRED).
```

Related Command

"HELP" on page 4-29	Requests help from a server FTP program. (By contrast, the **?** command requests help from the Client FTP)

ABOR

The **ABOR** command instructs the Server FTP to abort the previous command and any associated transfer of data. No action is taken by the Server FTP if the previous command has been completed (including data transfer). The control connection to the Server FTP is not closed, but the data connection is closed.

ABOR

Syntax Description

This command has no arguments or keywords.

Default

If no prefix is given, the **ABOR** request is directed to both Server FTPs.

Usage Guidelines

A host prefix is optional with the **ABOR** command. If a host prefix is specified, the **ABOR** request is directed only to the specified Server FTP.

Example

```
PUT 'scm.p015235.dump' tempdump
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru F   Mode S   Path SCM.P015235.DUMP
A:Volser SYSPK1   Unit SYSALLDA   Dsorg PS   Recfm VB   Lrecl 138
A:Blksize 23476   Rlse
A:150
B:150 ASCII data connection for tempdump (138.42.224.15,4127).
ABOR
B:552 tempdump: Connection reset by peer.
A:426-Data transfer aborted
A: 57344 bytes sent in 13.3 seconds (4295 bytes/s)   Path SCM.P015235.DUMP
A:User MYID   Data bytes sent 66334
A:Disk tracks read 1
A:426
B:225 ABOR command successful.
A:226 Abort command completed.
```

ADD

The **ADD** command requests that a file from host **A** be appended to a file at host B. The file retrieved is always at host A and the target file is always at host B. You might need to use the **A=B** command to switch host prefixes so that the correct host is associated with the correct host prefix. See the section on "A=B" on page 4-51 for details about the **A=B** command. See also the "PUT Example" on page 4-56 and "GET Example" on page 4-57 sections for examples of how the **A=B** command is used.

ADD [*path_name_a*] [*path_name_b*]

Syntax Description

path_name_a	File name of the file to be retrieved from host A.
path_name_b	File name on host B to which that file from host A is to be appended.

Default

If either file name is omitted, you are prompted for the file name.

Usage Guidelines

- The syntax for each path name depends on the associated Server FTP.
- A host prefix is not allowed with the **ADD** command.

Example

```
ADD n.d tempdump
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru F   Mode S   Path MYID.N.D
A:Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl  80
A:Blksize 3120   Rlse
A:150
B:150 ASCII data connection for tempdump (138.42.224.15,4128).
A:226-Transfer Complete
A: 3439 bytes sent in 2.49 seconds (1381 bytes/s)   Path MYID.N.D
A:User MYID   Data bytes sent 6480
A:Disk tracks read 1
A:226
B:226 ASCII Transfer complete.
```

ALLO

The **ALLO** (allocate) command provides a file size to those Server FTPs that require it. Refer to the documentation for your particular Server FTP to see if you need this command. The TCPaccess Server FTP does not require use of the **ALLO** command; it can be used optionally to cause records to be truncated or to allocate space.

ALLO integer1 [*R integer2*]

Syntax Description

integer1	Number of logical bytes of storage to be reserved for the file.
R integer2	Optional; specifies the maximum record or page size.

Usage Guidelines

- For files sent with record or page structure, a maximum record or page size (in logical bytes) can be required. This is indicated by the second integer argument.
- The *integer2* argument is optional, but if it is specified, it must be separated from the first by the three characters, “ **R** ” (space **R** space).
- A host prefix is required with the **ALLO** command.

Example

In the following example, the **ALLO** command truncates records to a length of 60 bytes during a file transfer between two TCPaccess systems.

```
STRU R
b:site lrecl(60) blk(6000)
b:allo 12000 r 60
PUT n.d allor.data
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru R   Mode S   Path MYID.N.D
A: Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB
A: Lrecl 80   Blksize 3120   Rlse
A:150
B:150-Data set open with attributes:
B:Type A N   Tabs 8   Stru R   Mode S   Recall 5
B:Path MYID.ALLOR.DATA
B:Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB
B:Lrecl 60   Blksize 6000   Space 2 1 Tracks Rlse   Maxr 60
B:150
A:226-Transfer Complete
A: 3441 bytes sent in 0.32 seconds (10753 bytes/s)  Path MYID.N.D
A:User MYID   Data bytes sent 6480
A:Disk tracks read 1
A:226
B:226-Transfer Complete
B: 3441 bytes received in 0.44 seconds (7820 bytes/s)
B:Path MYID.ALLOR.DATA   User MYID   Data bytes received 3277
B:Disk tracks written 1   Records truncated 21
B:Records padded 80   Records folded 20
B:226
```

BYE

The **BYE** command is the same as the **QUIT** command. Read “QUIT” on page 4-38 for details on the **BYE** and **QUIT** commands.

BYE

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

A host prefix is required with the **BYE** command.

Example

```
A:BYE
A:221 Goodbye.
```

Related Commands

“END” on page 4-26	Can be used instead of **BYE**; **END** does not require a host prefix.
“QUIT” on page 4-38	Can be used instead of **BYE**; **QUIT** requires a host prefix and takes no arguments.

CDUP

The **CDUP** command directs a Server FTP to change the current directory to the parent directory of the old current directory. The **CDUP** command is most useful when the Server FTP manipulates a hierarchical file system such as UNIX.

CDUP

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

A host prefix is required with the **CDUP** command.

Example

In the following example, a UNIX Server FTP in session with the Client FTP program has /u/user1/work as the current directory. If a **CDUP** command is issued by the Client FTP to that UNIX Server FTP, the resulting current directory is the parent of the old current directory (/u/user1).

```
B:CDUP
B:250 CWD command successful.
```

CONN

The **CONN** command sets up one control connection between the Client FTP program and a Server FTP. Since the TCPaccess Client FTP uses the third-party model, two such connections must be established, one to each Server FTP that participates in the file transfer. If the same Server FTP is used as both Server FTPs, two **CONN** commands are required. Since the **CONN** command sets up the control connections to the Server FTPs, issue the **CONN** commands first when using the Client FTP program.

CONN [*hostname*]

Syntax Description

CONN	Command used to establish connections between client programs and Server FTP.
hostname	Provides the host name to which one FTP control connection is established. If you omit the host name, you are prompted to provide one.

Default

If the host prefix is omitted, the first available unused host prefix is assigned to that connection.

Usage Guidelines

- Host name strings must correspond to the syntax specified in “Using Host Name Strings” on page 1-5.
- A host prefix is optional with the **CONN** command. If a host prefix is specified, the connection is assigned to that host prefix.

Example

```
CONN 10.1.1.99
A:220 MVS.HQ.COMPANY.COM -- FTP Server, Enter command or HELP
CONN unix
B:220 unix FTP server (SunOS 4.1) ready.
```

CWD

The **CWD** command requests that a Server FTP change the current directory being maintained for you by the Server to a new directory. Server FTP can access UNIX System Services (OpenEdition) files and MVS datasets.

CWD [*path_name*]

Syntax Description

path_name	Indicates to the Server FTP the name of the directory to be made the current directory.

Default

If you omit *path_name*, the Client FTP program prompts for one.

Usage Guidelines

- The syntax for *path_name* depends on the associated Server FTP.
- A host prefix is required with the **CWD** command.
- If the Server FTP **SITE** command does not specify the format (HFS or MVS), then Server FTP uses the change directory to determine the format. See Example 2.

Example 1

In the following example, a UNIX Server FTP in session with the Client FTP program has /u/user1/work as the current directory. If a **CWD junk** command is issued by the Client FTP to that UNIX Server FTP, the resulting current directory is the junk subdirectory of the old current directory (/u/user1/work/junk). The same result is achieved by specifying **CWD /u/user1/work/junk**.

```
B:CWD /u/myid/acces
B:250 CWD command successful.
```

Example 2

In this example, the directory you are changing to determines the format.

```
CWD /
```

Indicates an HFS file.

```
CWD 'xname'
```

Indicates and MVS dataset.

DELE

The **DELE** (delete) command directs a Server FTP to delete the specified file.

DELE [*path_name*]

Syntax Description

path_name	Specifies the specific file to delete.

Default

If you omit *path_name*, you are prompted to provide one.

Usage Guidelines

- The syntax for *path_name* depends on the associated Server FTP.
- A host prefix is required with the **DELE** command.

Example

```
A:DELE t1.data
A:250 Deleted OK.
```

DO

The **DO** (do TSO) command requests the Client FTP program to execute a TSO command for you.

DO *tso_command parameters*

Syntax Description

tso_command	TSO command followed by any parameters to be passed to the TSO command.

Usage Guidelines

- The **DO** command is handled by the Client FTP program.
- A host prefix is not necessary and is ignored if included with the command.

Note: In batch mode, the TSO environment is required for a **DO** command to work.

Example

```
A:DO listc l(myid)
IN CATALOG:CATALOG.MVSICF1.VMVSTSO
MYID.ACCES.ASM
MYID.LIB.LOAD
MYID.T.D
MYID.VBIG.D
MYID.VB.D.
```

END

The **END** command terminates the Client FTP program. This is typically the last command you enter. Any open control connections are closed before the program terminates.

END

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

- Any host prefix is ignored.

Related Commands

You can use the **BYE** and **QUIT** commands interchangeably with **END**. Both **BYE** and **QUIT** require a host prefix; **END** does not.

"BYE" on page 4-20	Can be used instead of **END**; **BYE** requires a host prefix.
"QUIT" on page 4-38	Can be used instead of **BYE**; **QUIT** requires a host prefix and takes no arguments.

EXPE

The **EXPE** command toggles the use of experimental or regular directory commands. Since there is no consistent support for this command, it is recommended that you not use this command.

Note: The directory commands were added to FTP subsequent to the initial FTP specification and are documented in ***RFC 959***, ***File Transfer Protocol (FTP)***, Appendix II, Directory Commands.

EXPE

Syntax Description

No operands are associated with the **EXPE** command, since it is a Client FTP command.

Usage Guidelines

- Any host prefix is ignored.

"Table 4-2" on page 4-27 shows the FTP command that is sent over the control connection for each directory command with an **EXPE** setting:

Table 4-2 Using Directory Commands with an EXPE Setting

Client FTP Command	Regular	Experimental
MKD	**MKD**	**XMKD**
RMD	**RMD**	**XRMD**
PWD	**PWD**	**XPWD**
CDUP	**CDUP**	**XCUP**

GET

The **GET** command requests that a file from host B be copied to a file on host A by the appropriate Server FTPs. The file to be retrieved is always at host B, and the file to be copied to is always at host A. You might need to use the **A=B** command to switch host prefixes to get the correct host. An alternative to switching host prefixes is to use the **PUT** command; read the "PUT" on page 4-36 section.

GET [*path_name_b*] [*path_name_a*]

Syntax Description

path_name_b	File name to be retrieved from host B.
path_name_a	File name at host A into which the file from host B is to be copied.

Default

If you omit either path name, you are prompted for one.

Usage Guidelines

- The syntax for each path name depends on the associated Server FTP.
- A host prefix is not allowed with the **GET** command.

Example

```
GET jclbr14 cntl(newbr14)
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru F   Mode S   Recall 5
A:Path MYID.CNTL(NEWBR14)
A:Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB
A:Lrecl 80   Blksize 3120   Space 31 15 Tracks Rlse   Dir 5
A:150
B:150 ASCII data connection for jclbr14 (138.42.128.13,4106) (810 bytes).
A:226-Transfer Complete
A: 820 bytes received in 3.24 seconds (253 bytes/s)
A:Path MYID.CNTL(NEWBR14)   User MYID   Data bytes received 800
A:Disk tracks written 1
A:226
B:226 ASCII Transfer complete.
```

HELP

The **HELP** command requests help information from one or both of the Server FTPs.

HELP [*text*]

Syntax Description

text Any command for which more information or usage guidelines are needed.

Default

If no host prefix is given, help is requested from each host with an open control connection.

Usage Guidelines

- **HELP** has no required arguments.
- Any operands specified on the **HELP** command are passed to Server FTP unchanged and are interpreted by Server FTP. Different Server FTPs may interpret these operands differently.
- A host prefix is optional with the **HELP** command. If a host prefix is specified, help information is requested only from that host.

Example

```
A:HELP REST
--- HELP ---
*** HELP REST ***
FTPREST (Restart) Command:
Function: Specifies that the data transfer command which follows
 (immediately) is to restart at a specified intermediate
 point in the file.
Syntax: REST <marker>
Notes:
  (1) After a REST command, STOR and APPE have identical meanings.
  (2) Data transfer must be MODE B (block).
  (3) A file RETRieved will normally include restart
      markers approximately every 32767 bytes.  The REST
      parameter on the SITE command allows you to change
      this interval or even entirely suppress restart markers.
      See HELP SITE.  The actual decision to send a marker
      depends on a count of data bytes read from the disk
      not including OS count/control bytes).  When this count
      reaches the limit, the marker is sent at the next end
      of a complete logical record, segment of a spanned record
      (if RECFM includes VS), or a physical disk block (if
      RECFM is U, V, or F).
  (4) FTP can accept (and send) restart markers in either
      STRU F or STRU R.
  (5) FTP restart markers consist of 10 characters, which
      are the hex representation of five 8-bit bytes: TTRBB.
      Here "TTR" forms a standard OS disk block address, and
      BB is a byte offset within the block.
214 <end of HELP>
```

Related Command

? command Requests help from the Client FTP program.

LIST

The **LIST** command requests a Server FTP to provide current information for files corresponding to a specified path name. This contrasts to the **NLST** command that provides only a list of file names without any other file information.

LIST [*path_name_a*] [*path_name_b*]

Syntax Description

path_name_a	Path name to be listed from host A.
path_name_b	File name at host B into which the list from host is copied.

Default

If you omit either path name, you are prompted to provide one.

Usage Guidelines

- If the path name specifies a directory or other group of files, the Server FTP transfers current information for a list of files.
- If the path name specifies a single file, the Server FTP transfers current information on that file.
- The syntax for each path name depends on the associated Server FTP.
- A host prefix is not allowed with the **LIST** command.

Example

```
LIST acces ibmcatent
A:125 Transfer started
B:150 ASCII data connection for ibmcatent (138.42.128.13,4107).

A:226-Transfer Complete
A: 432 bytres sent in 1.03 seconds (419 bytes/s)  Path MYID.ACCES
A:User MYID   Data bytes sent 420
A:226
B:226 ASCII Transfer complete.
```

LOG

The **LOG** (login) command identifies the user by sending a user ID and password to a Server FTP.

You may change your password when logging into the TCPaccess server. You typically issue the **LOG** command immediately following the **CONN** command.

LOG [*userid***] [***current_password***] /[***new_password***]**

Syntax Description

userid	User name.
current_password	User's current password.
new_password	New password to be applied to the account.

Default

If you omit either the *userid* or *current_password*, the Client FTP program prompts you for them.

Usage Guidelines

The */new_password* parameter is a 1 to 8 character string password. The new password replaces the current password after the user ID and current password are validated. The new password option is valid only when in a session with a TCPaccess server. The slash (/) must follow the current password without any intervening blanks. The new password must follow the slash without any intervening blanks.

- If the Server FTP requires additional accounting information during the user identification process, the Client FTP program prompts you to enter the accounting data.
- A host sprefix is required with the **LOG** command.

Example

```
B:LOG myid
331 Enter PASS command
Password:
```

In the following example, user USER01 changes his current password from CJAY to MACDUFF:

```
A:LOG user01 cjay/macduff
230 User USER01 logged in.
```

MKD

The **MKD** (make directory) command directs a Server FTP to create the specified directory. The exact FTP command sent to the Server FTP program depends on the setting of the **EXPE** variable. For details regarding the **EXPE** variable, see the “EXPE” on page 4-27.section

MKD [*path_name*]

Syntax Description

path_name	Directory to be created.

Default

If you omit *path_name*, you are prompted for it.

Usage Guidelines

- If the path name is relative, the specified subdirectory is created in the current working directory.
- If the path name is absolute, the specified directory is created.
- The syntax for *path_name* depends on the associated Server FTP.
- A host prefix is required with the **MKD** command.

Example

For an example of **MKD** operation, consider this case:

The current directory for a session between a UNIX Server FTP and a Client FTP program is /u/user1/work. If a **MKD junk** command is issued by the Client FTP to that UNIX Server FTP, the subdirectory **junk** is created as a subdirectory in the current directory (/u/user1/work/junk).

The same result can be achieved by specifying **MKD /u/user1/work/junk**.

```
B:MKD /u/myid/sampdir
B:257 MKD command successful.
```

MODE

The **MODE** command sets one of three transmission modes:

- Block Mode

Block mode formats the data and allows for restart procedures.

- Compressed Mode

Compressed mode compresses the data for efficient transfer.

- Stream Mode

Stream mode passes the data with little or no processing. It interacts with the structure attribute to determine the type of processing. Stream mode is the default if no **MODE** command was used.

The sending host translates its internal end-of-line or end-of-record representation into the representation required by the transfer mode and file structure; the receiving host performs the inverse translation to its internal representation. Because these transformations make extra work for some systems, identical systems transferring non-record structured text files might use binary representation and stream mode to simplify transfer.

MODE S | B | C

Syntax Description

MODE	Directory to be created.
S	Stream mode.
B	Block mode.
C	Compress mode.

Default

If the prefix is omitted, the **MODE** command is directed to each Server FTP with an open connection.

Usage Guidelines

- One of the three codes is required as an argument on the **MODE** command.
- A host prefix is optional with the **MODE** command. If a host prefix is specified, the **MODE** command is directed only to that Server FTP

Example

Each of the possible transmission modes is discussed in the following sections. For a detailed description of the effect of various transmission modes, read the “Transmission Modes” section in *RFC 959, File Transfer Protocol*.

Not all Server FTPs support all transmission modes; review the documentation for the target Server FTP if you have questions concerning transmission mode support.

Note: TCPaccess Server FTP does not support compressed mode. Client FTP uses the third-party model (server-to-server) and an environment can exist where TCPaccess's Server FTP is not implemented. In this instance, compress mode is valid if both remote servers support compress mode.

Block Mode

Set block mode with the character B. In block mode, the file is transmitted as a series of data blocks preceded by one or more header bytes. Record structures are allowed in this mode, and any representation type can be used. Restart markers are embedded in the data stream.

Compressed Mode

Set compressed mode with the character C. In compressed mode, filler bytes (space characters in ASCII or EBCDIC) and replicated data bytes are compressed when transmitted over the data connection. Compressed mode can increase bandwidth on very large network transmissions at little extra CPU cost. Additionally, compressed mode reduces the size of printer files.

Set the structure with the **STRU** command. In compressed mode, the representation type determines the filler byte. Set this with the **TYPE** command.

Stream Mode

Set stream mode with the character S. This is the default if no **MODE** command has been used. In stream mode, the data is transmitted as a stream of bytes. There are no restrictions on the representation type used, and record structures are allowed. In a record structured file, End of Record (EOR) and End of File (EOF) are each indicated by a two-byte control code included with the data sent over the data connection. If the structure is a file structure, the EOF is indicated by the sending host closing the data connection, and all bytes sent over the data connection are data bytes.

NLST

The **NLST** command directs a Server FTP to provide a list of file names in a specified directory or file group. That is, the Server FTP returns a list of file names with no additional information. This contrasts to the **LIST** command, which provides the file names and other current information about the files.

NLST [*path_name_a***] [***path_name_b***]**

Syntax Description

path_name_a	Path name to be listed from host A.
path_name_b	File name at host B into which the name list from host A is copied.

Default

If you omit either path name, you are prompted for it.

Usage Guidelines

- If the path name specifies a directory or other group of files, the Server FTP transfers a corresponding list of file names.
- If the path name specifies a single file, the Server FTP transfers the file name of that file.
- The syntax of each path name depends on the associated Server FTP.
- A host prefix is not allowed with the **NLST** command.

Example

```
NLST acces ibmcatent2
A:125 Transfer Started
B:150 ASCII data connection for ibmcatent2 (138.42.224.15,4137).
B:226 Transfer complete.
A:226-Transfer Complete
A: 70 bytes sent in 0.54 seconds (129 bytes/s)   Path MYID.ACCES
A:User MYID
A:Data bytes sent 58
A:226
```

PUT

The **PUT** command requests that the appropriate Server FTP copy a file from host A to a file on host B. The file to be copied is always at host A and the file destination is always at host B. You might need to use the **A=B** command to switch host prefixes and associate the correct host with the correct host prefix.

PUT [*path_name_a***] [***path_name_b***]**

Syntax Description

path_name_a	Name of the file to be copied from host A.
path_name_b	File name at host B into which the file from host A is copied.

Default

If you omit either path name, you are prompted for the file name.

Usage Guidelines

- The syntax for each path name depends on the associated Server FTP.
- A host prefix is not allowed with the **PUT** or **SEND** commands.

Example

```
PUT n.d put_example
A:150-Data set open with attributes:
A:Type A N   Tabs 8   STRU F   Mode S   Path MYID.N.D
A:Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB
A:Lrecl 80   Blksize 3120   Rlse
A:150
B:150 ASCII data connection for put_example (138.42.224.15,4138).

A:226-Transfer Complete
A: 3439 bytes sent in 2.12 seconds (1622 bytes/s)   Path MYID.N.D
A:User MYID   Data bytes sent 6480
A:Disk tracks read 1
A:226
B:226 ASCII Transfer complete.
```

Related Commands

“GET” on page 4-28	An alternative to switching host prefixes; read “GET” on page 4-28 for details
“SEND” on page 4-44	Can be used in place of **PUT**. There are no differences between **PUT** and **SEND.**

PWD

The **PWD** command directs a Server FTP to return the path name of the current working directory.

PWD

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

A host prefix is required.

Example

```
B:PWD
B:257 "/u/myid" is current directory.
```

QUIT

The **QUIT** command disconnects you from a host. It logs you out and terminates the connection between you and the Server FTPs. The **BYE** command is a synonym for the **QUIT** command. The **QUIT** command is the opposite of the **CONN** command.

QUIT

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

The **QUIT** command requires a prefix.

Example

```
A:QUIT
A:221 Session Terminated.
```

Related Commands

“BYE” on page 4-20	Can be used instead of **QUIT**. **BYE** works exactly as the **QUIT** command.
“END” on page 4-26	Can be used instead of **QUIT**; **END** does not require a host prefix.

QUOT

The **QUOT** (quote) command sends an uninterpreted, unaltered character string to the Server FTP over the control connection. This mechanism sends FTP commands to the Server that the Client FTP program might not be able to send.

QUOT [*text*]

Syntax Description

text	Sent to the Server over the control connection exactly as you enter it.

Default

If the text is omitted, you are prompted to enter it.

Usage Guidelines

A host prefix is required with the **QUOT** command.

Example

```
A:QUOT site norlse
```

REN

The **REN** (rename) command directs a Server FTP to rename a file.

REN [*old_path_name*] [*new_path_name*]

Syntax Description

old_path_name	File name to be renamed.
new_path_name	New name to be assigned to that file.

Default

If you omit either argument, you are prompted to enter it.

Usage Guidelines

- The syntax of the path names depends on the associated Server FTP.
- A host prefix is required with the **REN** command.

Example

```
A:REN t2.data rename.data
A:350 Requested file action pending further information
A:250 Renamed OK
```

REST

The **REST** (restart) command shows the Server FTP the restart marker where a file transfer is to be restarted.

This command does not cause a file transfer but instead causes the Server FTP to skip over the file to the specified data checkpoint. This command should be followed immediately by the Client FTP command that causes the file transfer to resume.

Note: The restart facility requires that you run in **MODE B**. Many UNIX implementations do not support **MODE B** and cannot use the restart facility.

REST *marker*

Syntax Description

marker	Server FTP marker point where the file transfer is to be restarted.

Usage Guidelines

- The marker is required in the restart command.
- The format of the restart marker is determined by the sending Server FTP and should be entered exactly as displayed during the interrupted file transfer.
- A host prefix is required with the **REST** command.

Example

The following is a restart marker message received by a user during a file transfer:

```
B:110 MARK 0100220040D82 = 0100220211C3
```

To restart the file transfer at the restart markers, issue the following commands to receive the following output:

```
TYPE i
MODE b
a:rest 010020040d82
b:rest 0100220211c3
a:site rest(100000)
put 'scm.p016572.t01tcp' psr16572.job
A:350 Requested file action pending further information
B:350 Requested file action pending further information
B:150-Data set open with attributes:
B:Type I N   Stru F   Mode B   Recall 5   Path MYID.PSR16572.JOB
B:Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 133
B:Blksize 6650   Space 3 1 Cyl Rlse   Restart at 0100220211C3
B:150
A:Type I N   Stru F   Mode B   Path SCM.P016572.T01TCP
A:Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 133
A:Blksize 6650  Rlse  Bytes/Restart 100000   Restart at 010020040D82
A:150

A:226-Transfer Complete
A: 326284 bytes sent in 3.39 seconds (96248 bytes/s)
```

```
A:Path SCM.P016572.T01TCP   User MYID   Data bytes sent 318934
A:Disk tracks read 9      Restart markers sent 3
A:226
B:110 MARK 010023010E8C = 010024060D3F
B:110 MARK 010025040F96 = 0100270408BB
B:110 MARK 0100280110A0 = 01002A02043B
B:226-Transfer complete
B: 326284 bytes received in 3.40 seconds (95965 bytes/s)
B:Path MYID.PSR16572.JOB  User MYID   Data bytes received 318934
B:Disk tracks written 9   Records folded 2472
B:Restart markers received 3
B:226
```

RMD

The **RMD** (remove directory) command directs a Server FTP to remove the specified directory.

RMD [*path_name*]

Syntax Description

path_name Directory to be removed.

Default

If you omit *path_name*, you are prompted for it.

Usage Guidelines

- If the path name is relative, the specified subdirectory is removed from the current working directory. If the path name is absolute, the directory is removed.
- The syntax of *path_name* depends on the associated Server FTP.
- A host prefix is required with the **RMD** command. Many systems require the directory to be empty before it can be removed.

Example

A UNIX Server FTP in session with the Client FTP program has /u/user1/work as the current directory. If an **RMD junk** command is issued by the Client FTP to that UNIX Server FTP, the junk subdirectory (/u/user1/work/junk) of the current directory is removed. The same result is achieved by specifying **RMD /u/user1/work/junk**.

```
B:RMD /u/myid/sampdir
B:250 RMD command successful.
```

SEND

The **SEND** command is the same as the **PUT** command.

SEND [*path_name_a*] [*path_name_b*]

Syntax Description

path_name_a	File name of the file to be copied from host A.
path_name_a	File name at host B into which the file from host A is copied.

Default

If you omit either path name, you are prompted for the file name.

Usage Guidelines

- The syntax for each path name depends on the associated Server FTP.
- A host prefix is not allowed with the **PUT** or **SEND** commands.

Example

```
SEND n.d send_example
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru F   Mode S   Path MYID.N.D
A:Volser COLPK1   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl  80
A:BLKSIZE 3120   Rlse
A:150
B:150 ASCII data connection for send_example (138.42.224.15,4148).
B:226 ASCII Transfer complete.
A:226-Transfer Complete
A: 3439 bytes sent in 4.66 seconds (737 bytes/s)   Path MYID.N.D
A:User MYID   Data bytes sent 6480
A:Disk tracks read 1
A:226
```

Related Commands

“GET” on page 4-28	An alternative to switching host prefixes.
“PUT” on page 4-36	Can be used in place of **SEND**. There are no differences between **SEND** and **PUT**.

SITE

The **SITE** (site parameters) command provides the Server FTP with specific information it requires. This information is essential to file transfers involving that Server FTP, but is not sufficiently universal to have been included specifically in the FTP. Typically, you use a **HELP SITE** Client FTP command to find the **SITE** requirements for a specific Server FTP. Otherwise, review the Server FTP documentation for the **SITE** requirements.

SITE *text*

Syntax Description

text	Passed through unchanged to the specified server.

Usage Guidelines

- Text is required in the **SITE** command syntax.
- A host prefix is required with the **SITE** command.

Detailed documentation of this command is in “SITE” on page 5-15.

Example

```
A:SITE vol(mvsts2)
```

SNDS

The **SNDS** (re-send **SITE** parameters) command directs the Client FTP program to re-send the last **SITE** command to the specified Server FTP. Your **SITE** command is reissued without your having to retype it. Since most Server FTPs require that new site parameters be provided before each data transfer, the **SNDS** saves time if identical site parameters are to be used repeatedly.

SNDS

Syntax Description

This command has no arguments or keywords.

Usage Guidelines

A host prefix is required with the **SNDS** command.

Example

```
A:SNDS
site vol(mvsts2) <SENT
```

STAT

The **STAT** (status) command requests a status response from a specified server.

STAT [*path_name*]

Syntax Description

path_name	Optional; specifies path to server.

Default

If no path name is given, the indicated Server FTP sends status information relative to parameters and connection status.

Usage Guidelines

- When **STAT** is issued between data transfer operations, the path name argument can be given. When it is given, the command works the same as the **LIST** command: It displays current information about the referenced files.

  When using the **STAT** command, the information is transferred over the control connection instead of the data connection.

- The syntax of *path_name* depends on the associated Server FTP(s).
- A host prefix is optional with the **STAT** command
  - If a host prefix is specified, the **STAT** command is sent only to the specified host.
  - If the prefix is omitted, the **STAT** command is sent to both hosts that have open connections.
- The TCPaccess Server FTP program implements some additional parameters on the **STAT** command. Use a **HELP STAT** Client FTP command to find additional parameters.

Example

In the following example, the status for the A host is requested.

```
a:STAT *
A:211--- STATUS ---
A:   -- FTP Parameters --
A:Remote DT Host, Port 138.42.32.160, 0
A:Local DT  Host, Port 138.42.224.15, 0
A:Type A N   Tabs 8   Stru F   Mode S   Recall 5   Server is passive
A:   -- END --
A:   -- Control --
A:User MYID   Acct Accs E0000200   Unit SYSALLDA   Host 138.42.224.15
A:   -- End Control --
A:   -- Path Data --
A:Rlse
A:   -- End Path Data --
A:   -- Transfer Information --
A:Data Transfer not in progress   Data bytes sent 6480
A:Disk tracks read 1   Network bytes sent 3439   Elapsed time 00.00.04
A:Bytes/Second 737
A:   -- END --
A:211 <End of Status>.
```

STRU

The **STRU** (file structure) command provides information on file structure to a Server FTP.

STRU F | R

Syntax Description

F	Specifies the file structure. File structure is used for files with no internal structure, and the file is considered to be a contiguous sequence of data bytes.
R	Specifies the record structure. This is for files made up of sequential records. Record structure is accepted for text files (such as, files with type ASCII or EBCDIC) by all FTP implementations.

Defaults

If no **STRU** command has been used, the default file structure is **F**.

If no host prefix is specified, the command goes to both Server FTPs.

Usage Guidelines

- One argument is required on the **STRU** command to set the file structure.
- A host prefix is optional with the **STRU** command. When a host prefix is given, the **STRU** command goes only to the specified Server FTP.

TYPE

The **TYPE** command tells a Server FTP the data type to use.

TYPE I | **L** *byte_size* | {**A** | **E** [**N** | **T** | **C**]}

Syntax Description

I	Image type. The data is sent as a contiguous bit stream that, for transfer, is packed into 8-bit transfer bytes. The receiving site stores the data as contiguous bits. The receiving storage system might need to pad the file (or each record, in record-structured files) to some convenient boundary. Review the documentation for a Server FTP to find out about padding. Image type is for the efficient storage and retrieval of files and for transfer of binary data. All FTP implementations are required to support the image type.
L *byte_size*	Local file type and the logical byte size of the file. The byte size value (*byte_size*), representing the logical byte size, is required with the local type. With this type, the data is transferred in logical bytes of the specified size. The logical byte size might differ from the transfer byte size. If the logical and transfer byte sizes differ, the logical bytes are packed contiguously disregarding transfer byte boundaries and are padded at the end if necessary. When the data reaches the receiving host, it is transformed in a manner dependent on the logical byte size and the particular host. The transformation is invertible; an identical file can be retrieved if the same parameters are used.
A	Sets the file type to ASCII. This type is accepted by all FTP implementations and is good for transferring text files, except when both hosts find the EBCDIC type more convenient. In accordance with the *NVT* standard, the **CRLF** sequence is used at the end of a line of text. The sender converts the data from an internal character representation to the standard 8-bit NVT ASCII representation (see the TELNET specification in the list of reference documents). The receiver converts the data from this standard form to the receiver's own internal form.
E	Sets the file type to EBCDIC, which performs efficient transfer between hosts that use EBCDIC. TCPaccess Client FTP2 users usually use this type when copying files to their MVS host. Data is transmitted as 8-bit EBCDIC characters. The character code is the only difference between EBCDIC and ASCII types. End-of-line is rarely used with EBCDIC type to denote structure, but where it is necessary, the **NL** character is used. The types ASCII and EBCDIC optionally take a second parameter that indicates what kind of vertical format control, if any, is associated with a file. If a file is to be sent to a host for printing, the vertical format control must be defined in the format expected by the target. The ASCII and EBCDIC types have a second parameter specifying non-print, TELNET, or carriage control (ASA).
N	**N** sets non-print format control. This is used when the file does not contain vertical format information. Normally, this format is used with files destined for processing or for storage. Non-print format is accepted by all FTP implementations.
T	**T** sets TELNET format control. This is used when the file contains ASCII/EBCDIC vertical format controls (i.e., **CR**, **LF**, **NL**, **VT**, **FF**). The characters **CRLF**, in exactly this sequence, also denote end-of-line.

C **C** sets carriage control (ASA) format control. This is used when the file contains ASA (FORTRAN) vertical format control characters.

ASA standard specifies these control characters:

blank	Move paper up one line.
0	Move paper up two lines.
-	Move paper up three lines.
1	Move paper to top of next page.
+	No movement (that is, type will overprint)

Default

For both ASCII and EBCDIC file types, vertical format control **N** is the default.

ASCII is the default argument for the **TYPE** command.

Non-print format is the default.

Usage Guidelines

- One of the four arguments (**I**, **L** *byte_size*, **A**, or **E**) is required.
- If local type (**L**) is set, the integer byte size argument must also be set.
- If ASCII (**A**) or EBCDIC (**E**) type is set, one of the three vertical format control arguments, **N**, **T**, or **C**, also can be set.

A=B

The **A=B** (switch prefixes) command directs the Client FTP program to switch the host prefixes associated with each Server FTP connection. This command gets the correct host associated with the correct host prefix before issuing a data transfer command such as **ADD**, **NLST**, or **LIST**.

A=B

Syntax Description

This command has no arguments or keywords.

Default

The **A=B** command is directed to the Client FTP program and any host prefix is ignored.

Example

The A host is an MVS system and the B host is a UNIX system. You want to obtain a name list of the current directory on the UNIX system. If you issue the Client FTP **NLST** command to obtain a name list, the **NLST** command obtains the name list from the **A** host and copies it to a file on the **B** host. Since the UNIX host is the B host, issue the **A=B** command to make the UNIX host the A host before issuing the **NLST** command.

A?B

The **A?B** (show connections) command displays the current connections associated with the **A** and **B** host prefixes from the Client FTP program.

A?B

Syntax Description

This command has no arguments or keywords. The following output is produced by the **A?B** command:

```
HOST - A:host_name_a
HOST - B:host_name_b
```

Usage Guidelines

- The host names displayed are the host names in the **CONN** commands. If a connection is not established for a host prefix, the response is NOT CONNECTED.
- The **A?B** command goes to the Client FTP program.
- A host prefix is not necessary with the **A?B** command and is ignored.

Example

A?B

```
HOST - A:MVS
HOST - B:UNIX
```

Restart Support

If a file transfer is interrupted, it can be restarted. However, restart support requires that **MODE B** be specified.

The restart marker provided by TCPaccess is six bytes long in the format *VTTRBB*. *V* is the volume sequence number, **TTR** is the standard IBM OS disk block address, and *BB* is a byte offset within the block.

How to Restart

Use the **SITE REST**(*XXXXX*) FTP command to tell a TCPaccess FTP Server how often to send a restart marker. Send a restart marker after the sending of the record that exceeds or equals *XXXXX* number of bytes (varying between 1 and 500,000). Send the **SITE** command only to the **RETR** side of a data transfer. Other FTP Servers may initiate sending restart markers in a different way from TCPaccess.

A 110 message is written once per output block if a restart marker is sent somewhere in the data written for the block.

For example, suppose during a data transfer this restart mark message is sent:

```
B:110 MARK 0100030212C0 = 010003020FA0
```

You can restart the transfer at this point by sending the first number to the **RETR** side and the second number to the **STOR**. If you receive this restart mark message during an aborted data transfer, you can restart the transfer at these disk locations with these Client FTP **REST** commands:

```
to the RETR side: REST 0100030212C0
to the STOR side: REST 010003020FA0
```

TCPaccess supports restart markers (set at default value of every 32767 data bytes) if these conditions exist:

- **TYPE I**
- **MODE B**

Client FTP File Transfer Examples

This section provides some examples of file transfers. These examples illustrate use of the more frequently used commands and FTP features. They include:

- "PUT Example" on page 4-56
- "GET Example" on page 4-57
- "Transferring and Using a File in a Single JCL Job" on page 4-58
- "Restart File Transfer Example" on page 4-60
- "Managing Directories on UNIX-based Systems" on page 4-63

These examples follow the conventions described here.

FTP Invocation and Conventions

- Issue FTP under TSO to enter Client FTP.
- When Client FTP is ready for command input from a user, it places the FTP: prompt on the left side of the screen.

Host Prefixes

- Client FTP is based on the three-party model. The FTP protocols require that a user connect to a host before most commands can be issued. By convention, the first host the user connects to is the A side. Also by convention, the second host connected to is the B side.
- Commands that are specific to one host must be prefixed with the side identification (A or B) followed by a colon. Thus, to send a **SITE VOL** (MVSTSO) command to the A side, enter this command:

```
A:SITE VOL(MVSTSO)
```

- All command output sent from a host is prefixed by the host site ID and a colon. All the A side host output is prefixed by A:.

Successful Completion of a Transfer

- If the only response from the Client FTP command is **USERFTP**:, the command completed successfully.

Entering Text

- Text can be entered in uppercase or lowercase. Some host systems allow a mixture of lowercase and uppercase letters, while other host systems use uppercase for most functions. All commands entered are translated to uppercase before being sent to the servers. The data associated with a command is sent to its appropriate FTP server without case translation. The TCPaccess FTP server translates user IDs, passwords, data set names, and similar items to uppercase before the commands associated with them are executed.

Readability

- To improve readability, blank lines have been inserted between commands in the examples in this section. These lines do not appear in a real FTP terminal session.

PUT Example

In this FTP session, the **PUT** command transfers a file from host MVS host UNIX.

```
TCPaccess Rn Client FTP - Enter command or '?'
conn mvs
A:220 MVS.HQ.COMPANY.COM -- FTP Server, Enter command or HELP
a:log myid
331 Enter PASS command
Password:
conn unix
B:220 unix FTP Server (SunOS 4.1) ready.
b:log myid
331 Enter PASS command
Password:
put cntl(iefbr14) jclbr14
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru F   Mode S   Path MYID.CNTL(IEFBR14)
A:Volser COLPK1   Unit SYSALLDA   Dsorg PO   Recfm FB   Lrecl 80
A:Blksize 3120   Rlse
A:150
B:150 ASCII data connection for jclbr14 (138.42.224.15,4151).
A:226-Transfer complete
A: 820 bytes sent in 0.50 seconds (1640 bytes/s)  Path MYID.CNTL(IEFBR14)
A:User MYID   Data bytes sent 800
A:Disk tracks read 1
A:226
B:226 ASCII Transfer Complete
end
```

- The **conn mvs** command connects to host MVS. The first connection is to the A side. The response "A:220" on the next line shows that the connect command succeeded.
- The command **a:log myid** logs a user on to the A side. myid is the user ID. FTP prompts for a password, then sends the user ID and password combination to the A side host for validation. No error message is received and the USERFTP: prompt displays on the next line, indicating that the log command worked and user myid is logged on to the A host, MVS.
- The **conn unix** command connects to the B side host, unix.
- The command **b:log myid** logs a user onto the B side. FTP prompts for a password in a nondisplay field. FTP sends the userid/password combination to host unix. These are valid and user myid logs on to the B side.
- The command **put cntl(iefbr14) jclbr14** tells Client FTP to transfer file MYID.CNTL(IEFBR14) from the A side host, MVS. (FTP prefixes the user ID to the beginning of any non-quoted data set name referenced while a user is connected to an MVS host with TCPaccess running.) FTP creates or overwrites file jclbr14 on the side B host, unix.

  The A:226 message from the A side host means that all of file MYID.CNTL(IEFBR14) successfully transferred from the A side host, MVS. The B:226 messages mean that FTP successfully created or overwrote file jclbr14 on the B side host, unix.

- The **end** command ends the FTP session.

GET Example

In the following FTP session, the **GET** command transfers a file from host unix to host MVS.

```
TCPaccess Rn Client FTP - Enter command or '?'
conn mvs
A:220 MVS.HQ.COMPANY.COM -- FTP Server, Enter Command or HELP
a:log myid
331 Enter PASS command
Password:
conn unix
B:220 unix FTP server (SunOS 4.1) ready.
b:log myid
331 Enter PASS command
Password:
get jclbr14 cntl(newbr14)
A:150-Data set open with attributes:
A:Type A N   Tabs 8   Stru F   Mode S   Recall 5
A:Path MYID.CNTL(NEWBR14)
A:Volser COLPK1   Unit SYSALLDA   Dsorg PO   Recfm FB   Lrecl 80
A:Blksize 3120   Space 31 15 Tracks Rlse
A:150
B:150 ASCII data connection for jclbr14  (138.42.224.15,4098) (810 bytes).
A:226-Transfer Complete
A: 820 bytes received in 3.36 seconds (244 bytes/s)
A:Path MYID.CNTL(NEWBR14)   User MYID   Data bytes received 800
A:Disk tracks written 1
A:226
B:226 ASCII Transfer complete.
end
```

- The command **conn mvs** connects to host MVS. The first connection is to side A. The response A:220 on the next line shows that the connect command succeeded.
- The command **a:log myid** logs a user onto the A side. myid is the user ID. FTP prompts for a password, then sends the user ID and password combination to the A side host for validation. No error message is received and the USERFTP: prompt displays on the next line, indicating that the log command worked and user myid is logged on to the A side host, MVS.
- The command **conn unix** connects to the B side host, unix.
- The command **b:log myid** logs a user on to the B side. FTP prompts for a password in a nondisplay field. FTP sends the userid/password combination to host unix. These are valid and user myid logs on to the B side.
- The command **get jclbr14 cntl(newbr14)** tells Client FTP to transfer file jclbr14 from the B side host, unix. FTP creates or overwrites file MYID.CNTL(NEWBR14) on the A side host, MVS. (FTP prefixes the user ID to the beginning of any non-quoted data set name referenced when a user is connected to an MVS host with TCPaccess running.)

  The B:226 message from the B side host means that all of file jclbr14 successfully transferred from the B side host, unix. The A:226 message means that FTP successfully created or overwrote file MYID.CNTL(NEWBR14) on the A side host, MVS.
- The **end** command ends the FTP session.

Transferring and Using a File in a Single JCL Job

Transferring a file in one job step and using that file in another job step can run into a file allocation problem, wherein the file transfer fails in the FTP2 job step. This is usually because the file has been allocated to the batch job and the TCPaccess base product cannot allocate the file for a data transfer. The FTP2 and FTP client programs do not perform file transfers in their own address space; the FTP2 and FTP clients direct the TCPaccess base product to perform file transfers.

As a workaround, perform an IDCAMS ALTER NEWNAME of the file between the file transfer step and the use of the file.

Examples

The following sample JCL does a file transfer, performs an IDCAMS ALTER NEWNAME, and uses the file in the final job step (the data set MVS.P25206.DATA is created in the first job step, renamed to the file MVS.NEWNAME.DATA in the second step, and the file name MVS.NEWNAME.DATA is used in the last step):

```
//MVSA JOB (TSO00...99),'FTP2 BATCH',MSGCLASS=X,NOTIFY=MVS,CLASS=A
//*
//*  JOB TO TRANSFER A FILE AND THEN USE IT IN A LATER JOBSTEP
//*
//*  STEP TO DO THE FTP TRANSFER
//*
//FTP1        EXEC PGM=IKJEFT01,REGION=4000K
//SYSTSIN     DD *,DCB=BLKSIZE=80
    FTP2 / APP=ACCES  TEST NETRC
//STEPLIB     DD DCB=BLKSIZE=32000,
//               DISP=SHR,DSN=T01TCP.LINK
//SYSTSPRT    DD SYSOUT=X
//SYSPRINT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPUT      DD SYSOUT=*,DCB=BLKSIZE=133
//SYSVLT      DD SYSOUT=*,DCB=BLKSIZE=133
//NETRC       DD DISP=SHR,DSN=MVS.FTP.NETRC
//SYSGET      DD *,DCB=BLKSIZE=80
open unix
get temp 'mvs.p25206.data'
END
/*
//*
//*   STEP TO PERFORM AN IDCAMS ALTER NEWNAME
//*
//STEP2       EXEC PGM=IDCAMS
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD *
 ALTER 'MVS.P25206.DATA'                NEWNAME('MVS.NEWNAME.DATA') -
  CAT(CATALOG.TSO.VESA001)
//*
//*   THIS STEP PRINTS THE FILE ON THE SYSUT1 DD CARD
//*
//STEP3       EXEC PGM=IEBGENER
//SYSUT1      DD DISP=SHR,DSN=MVS.NEWNAME.DATA
```

```
//SYSUT2      DD SYSOUT=X,COPIES=1,DCB=*.SYSUT1
//SYSPRINT    DD   SYSOUT=*
//SYSIN       DD   DUMMY
```

The following sample JCL uses a file, performs an IDCAMS ALTER NEWNAME, and does a file transfer in the final job step (the file MVS.OLDNAME.DATA is used in the first jobstep, renamed to the file MVS.TRANSFER.DATA in the second jobstep, and the file MVS.TRANSFER.DATA is transferred in the last jobstep):

```
//MVSB JOB (TSO00,,,99),'FTP2 BATCH',MSGCLASS=X,NOTIFY=MVS,CLASS=A
//*
//* JOB TO TRANSFER A FILE AND USE IT IN A LATER JOBSTEP
//*
//*   THIS STEP PRINTS THE FILE ON THE SYSUT1 DD CARD
//*
//STEP1     EXEC PGM=IEBGENER
//SYSUT1    DD DISP=SHR,DSN=MVS.OLDNAME.DATA
//SYSUT2    DD SYSOUT=X,COPIES=1,DCB=*.SYSUT1
//SYSPRINT  DD    SYSOUT=*
//SYSIN     DD    DUMMY
//*
//*   STEP TO PERFORM AN IDCAMS ALTER NEWNAME
//*
//STEP2     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 ALTER 'MVS.OLDNAME.DATA'                NEWNAME('MVS.TRANSFER.DATA') -
  CAT(CATALOG.TSO.VESA001)
//*
//*  STEP TO DO THE FTP TRANSFER
//*
//FTP3      EXEC PGM=IKJEFT01,REGION=4000K
//SYSTSIN   DD *,DCB=BLKSIZE=80
FTP2 / APP=ACCES  TEST NETRC
//STEPLIB   DD DCB=BLKSIZE=32000,
//          DISP=SHR,DSN=T01TCP.LOAD
//SYSTSPRT  DD SYSOUT=X
//SYSPRINT  DD SYSOUT=*,DCB=BLKSIZE=133
//SYSPUT    DD SYSOUT=*,DCB=BLKSIZE=133
//SYSVLT    DD SYSOUT=*,DCB=BLKSIZE=133
//NETRC     DD DISP=SHR,DSN=MVS.FTP.NETRC
//SYSGET    DD *,DCB=BLKSIZE=80
open unix
put 'mvs.transfer.data' temp
END
/*
```

Transfer to an MVS Internal Reader

The following is an example of an FTP file transfer from a data set on host **unix** to an MVS internal reader on host **MVS**.

```
TCPaccess Rn Client FTP - Enter command or '?'
```

```
conn unix
A:220 unix FTP server (SunOS 4.1) ready.
a:log myid
331 Enter PASS command
Password:
conn mvs
B:220 MVS.HQ.COMPANY.COM -- FTP Server, Enter command or HELP
b:log myid
331 Enter PASS command
Password:
b:site submit
put jclbr14 anyname.data
B:150-Data set open with attributes:
B:Type A N   Tabs 8   Stru F   Mode S   Intrdr   Recfm FB   Lrecl 80
B:Blksize 20000
B:150
A:150 ASCII data connection for jclbr14 (138.42.224.15,20) (810 bytes).
A:226 ASCII Transfer complete.
B:226-Transfer complete
B: 820 bytes received in 0.30 seconds (2733 bytes/s)   User MYID
B:Data bytes received 800
B:226
end
```

- The command **conn unix** connects to host unix. The first connection is to the A side. The A:220 message on the next line shows that the connect succeeded.
- The command **a:log myid** logs a user on to the A side. The myid is the user ID. FTP prompts for a password. FTP sends the user ID and password combination to the A side host for validation. No error message is received and the USERFTP: prompt displays on the next line, indicating the log command succeeded. User myid is logged on to the A side host, unix.
- The command **conn mvs** connects to the B side host, MVS.
- The command **b:log myid** logs a user on to the B side. FTP prompts for a password in a nondisplay field. FTP sends the userid/password combination to host MVS. These are valid and user MYID logs on to the B side.
- The command **b:site submit** is to the B side host, MVS, directing the next data transfer to the MVS internal reader for execution. The **SITE** command is relevant to an MVS host where TCPaccess is running.
- The **put jclbr14 anyname.data** commands FTP to transfer file jclbr14 from the A side host, unix. Due to the previous command in step 5, the file is transferred to an MVS internal reader on side B, host MVS. The file name (*anyname.data*) for the B side host is ignored because no data set is being created or updated.
- The **end** command ends the FTP session.

Restart File Transfer Example

This is an example of a restart file transfer.

```
TCPaccess Rn Client FTP - Enter command or '?'
conn mvs
A:220 MVS.HQ.COMPANY.COM -- FTP Server, Enter Command or HELP
conn mvs2
```

```
B:220 MVS2.HQ.COMPANY.COM -- FTP Server, Enter Command or HELP
a:log myid
331 Enter PASS command
Password:
b:log myid
331 Enter PASS command
Password:
type i
mode b
a:site rest(100000)
b:site lrecl(133) recfm(vb) blksize(6650) space(3 1) CYL
put 'scm.p016572.t01tcp' psr16572.t01tcp
A:150-Data set open with attributes:
A:Type I N   STru F   Mode B   Path SCM.P016572.T01TCP
A:Volser HAGCAT   Unit SYSALLDA   Dsorg PS   Recfm FB   Lrecl 133
A:Blksize 6650   Rlse   Bytes/Restart 100000
A:150
B:150-Data set open with attributes:
B:Type I N   Stru F   Mode B   Recall 5   Path MYID.PSR16572.T01TCP
B:Volser ICSPK3   Unit SYSALLDA   Dsorg PS   Recfm VB   Lrecl 133
B:Blksize 6650   Space 3 1 Cyl Rlse
B:150
B:110  MARK  000204010A = 00020414F9
B:110  MARK  0005010214 = 0005021075
B:110  MARK  000704031E = 0007060BF1
B:110  MARK  000A010428 = 000A040771
B:110  MARK  000C040532 = 000D0202ED
B:110  MARK  000F01063C = 000F0517DE
B:110  MARK  0011040746 = 001203135E
B:110  MARK  0014010850 = 0015010EDA
B:110  MARK  001604095A = 0017050A56
B:110  MARK  0019010A64 = 001A0305D6
B:110  MARK  001B040B6E = 001D010152
B:110  MARK  001E010C78 = 001F041643
B:110  MARK  0020040D82 = 00220211C3
B:110  MARK  0023010E8C = 0024060D3F
B:110  MARK  0025040F96 = 00270408BB
B:110  MARK  00280110A0 = 002A02043B
A:226-Transfer Complete
A: 1656466 bytes sent in 52.1 seconds (31781 bytes/s)
A:Path SCM.P016572.T01TCP   User MYID   Data bytes sent 1619142
A:Dist tracks read 41   Restart markers sent 16
A:226
B:226-Transfer complete
B: 1656466 bytes received in 52.1 seconds (31775 bytes/s)
B:Path MYID.PSR16572.T01TCP    User MYID   Data bytes received 1619142
B:Disk tracks written 43   Records folded 12551
B:Restart markers received 16
B:226
end
```

- The two **conn mvs** commands connect to host MVS on the A side, and to host MVS2 on the B side.
- The **a:log myid** command logs user MYID onto the A side, and user MYID onto the B side.

- The **type i** command sets image type to a binary transfer.
- The **mode b** command sets block mode. Restart markers are embedded in the data stream.
- The **a:site rest(100000)** command tells the sending side how often to insert restart markers into the data.
- The **b:site lrecl(133) recfm(vb) blksize(6650) space(3 1) CYL** command changes some of the default allocation parameters on the B side.bu
- The **put** command tells FTP to transfer a file from the A side host, MVS. FTP creates file MYID.PSR16572.T01TCP on the side B host, MVS2.
- The A:150 messages give details of the files being transferred.
- The B:110 MARK messages mark blocks sent with embedded restart markers. The first number of the 110 message is the restart marker for the RETR side (the A side in this example). The second number of the 110 message is a restart marker for the STOR side (the B side in this example).
- The A:226 messages from the A side host indicate that all of file SCM.P016572.T01TCP successfully transferred from the A side host, MVS. The B:226 messages indicate that FTP successfully created file MYID.PSR16572.T01TCP on the B side host, MVS2.
- The **end** command ends the FTP session.

Managing Directories on UNIX-based Systems

These commands show how to manage directories on UNIX systems.

```
TCPaccess RnClient FTP - Enter command or '?'
conn unix
A:220 unix FTPserver (SunOS 4.1) ready.
a:log demo ftptest
a:pwd
A:257 "/u/demo" is current directory.
a:mkd tempdir
A:257 MKD command successful.
a:pwd
A:257 "/u/demo" is current directory.
a:cwd tempdir
A:250 CWD command successful.
a:pwd A:257 "/u/demo/tempdir" is current directory.
a:cdup
A:250 CWD command successful.
a:pwd
A:257 "/u/demo" is current directory.
a:rmd tempdir
A:250 RMD command successful.
a:pwd
A:257 "/u/demo" is current directory.
end
```

- The command **conn unix** connects the A side to host unix.
- The command **a:log demo ftptest** logs User demo on to the A side host unix.
- The command **a:pwd** asks the server to print the A side path name of its current directory.
- The command **a:mkd tempdir** asks the A side server to create a directory called tempdir.
- The command **a:cwd tempdir** changes the A side directory from /u/demo to /u/demo/tempdir.
- The command **a:cdup** changes the A side directory from its current directory /u/demo/tempdir to its parent directory u/demo.
- The command **a:rmd tempdir** asks the A side server to remove directory /u/demo/tempdir.
- The **end** command terminates the FTP session.

5

Server FTP

This chapter describes the server program for FTP within TCPaccess. It contains these sections:

- “Introducing Server FTP” on page 5-2

  Provides a brief overview of the Server FTP.

- “File Handling by the Server FTP” on page 5-3

  Describes how the Server FTP handles files.

- “Server FTP Commands” on page 5-7

  Describes the Server FTP commands and includes a table with brief descriptions for quick reference.

  “ALLO” on page 5-9 “HELP” on page 5-10 “MKD” on page 5-12 “REST” on page 5-13

  “RMD” on page 5-14 “SITE” on page 5-15 “STAT” on page 5-27

- “Data Set Attributes” on page 5-28

  Describes the attributes of the data sets that can be read or written by the Server FTP.

- “Data Transfer Operations” on page 5-39

  Describes the commands (**STOR**, **APPE**, and **RETR**) that initiate data transfer.

- “Character Type Rules” on page 5-42

  Describes the transformation rules for creating and retrieving character-type files.

- “Binary-Type Rules” on page 5-49

  Describes the transformation rules for creating and retrieving binary-type files.

- “Non-Invertible Retrieval” on page 5-51

  Describes some of the changes Server FTP makes to a file that might not be able to be undone when the file is retrieved.

- “Other Features” on page 5-52

  Lists some other useful features of the FTP program.

Introducing Server FTP

The Server FTP supports large-scale remote computing on a large IBM (and compatible) mainframe. It includes these types of support:

- File System Support

  The Server FTP supports creation and retrieval of a subset of the disk formats provided by the file system of MVS.

- Record Structure Support

  The MVS file system is record oriented. The FTP Server can translate character files between record structure locally and file structure remotely.

- Binary Files Support

  The Server FTP can transfer large files of binary data efficiently. The parameters required for record-structured binary files (STRU R, MODE B, and TYPE I) are implemented. Restart markers support restarts of large file transfers.

- UNIX System Services (OpenEdition) Support

  Server FTP lets you access UNIX System Services files on machines running TCPaccess. You may specify MVS datasets or UNIX System Services files using the SITE command. If you are not using the SITE command, Server FTP uses the directory to which you are changing to determine the format.

- JES Internal Reader Support

  Server FTP lets data transfer to MVS be submitted as a batch job to MVS via the JES Internal Reader facility (see the section of this chapter on procedures for JES Internal Reader support).

File Handling by the Server FTP

The Server FTP can read an existing disk data set with a wide variety of disk formats and map it correctly into the specified FTP parameters for transmission across the network. The record-structured MVS file system forces you to set limits on the size of a record when a file is created. Many processors require this record limit to be a “card image” (80 characters). A source file prepared on a stream-oriented system and transferred to the mainframe can contain records that are too long, and you may want to specify a larger record size with the **SITE** command.

Handling a Record Which is Too Long

When the Server FTP receives a file and finds a record too long, it does not discard data; it folds it into multiple records and informs you of its action. As each source language has a different continuation convention, folding the data in this manner is unlikely to match any of these conventions. When the Server FTP preserves data in this manner, you can easily fix the error later.

Individual warning messages are not issued for folded records. The Server FTP counts records folded and sends that count at the end of file transfer (if it is non-zero).

You can also have the Server FTP truncate rather than fold with the **ALLO R** command. See “ALLO” on page 5-9 and “SITE” on page 5-15 for details.

Transferring Files to a Host

The FTP lets a character file be transferred to a host for one of three purposes: for printing, for storage and later retrieval, or for processing. Under MVS, each of these purposes requires a different file format that must be chosen when the file is created.

- By default, the Server FTP assumes “processing” and records the data in a format that is likely to be acceptable to most MVS processing programs (a card image data set).
- When doing a **STOR** into a data set, the Server FTP infers the purpose from the FTP parameters and disk parameters and performs the appropriate translations.
- To create a print file, a print type must be specified (TYPE AT, TYPE ET, TYPE AC, or TYPE EC).

The translations the Server FTP performs for printing or processing are not exactly *invertible* if the file is later retrieved with FTP. If you want information to be stored for later retrieval in exactly the same form, you must override the default parameters with the **SITE** command.

Note: An exact representation of the network file is called *raveled*. Raveled files are invertible, meaning you can FTP them back to the originating operating system in exactly the same format they started with. Specify a raveled file when you want to store the file on MVS for later retrieval to the originating operating system. In most cases, a raveled file cannot be passed as input to any IBM processing program or sent to the printer.

For a detailed description of raveled and non-invertible files, see “Non-Invertible Retrieval” on page 5-51.

As an aid in setting data set attributes, canned attribute sets for the most common cases can be chosen by mnemonic name on the **SITE** command. For example, **SITE PRINT** chooses appropriate attributes for a print file.

Sophisticated File Handling

An experienced user can use additional facilities for more control over Server FTP operations. You can supply explicit Data Control Block (DCB) parameters in the **SITE** command to cause the FTP Server to create any reasonable sequential data set format used by MVS (the Server FTP may also create unreasonable formats).

Also, the Server FTP's default formats for creating new data sets are those generally used by MVS programs. You can override these formats to obtain information-conserving storage on the mainframe. The Server FTP lists the full MVS data set attributes as well as the FTP transfer parameters when a transfer starts and reports full statistics when the transfer completes.

Transferring Files To a Tape

The FTP server supports file transfer to and from magnetic tape volumes. This option can be specified dynamically with the FTP Server SITE command described in this chapter.

Configuration

To use this facility, you will need to set the parameters LABEL and MOUNT for the FTP statement in your APPCFGxx member of the PARM data set.

To provide installation defaults for tape data set allocation, the system administrator should set up a special GAT TYPE(TAPE) entry in the Generic Attributes Table (GAT) in APPCFGxx. Parameters of interest are COMPACT, LABEL(), PARALLELMOUNT, PRIVATE, and UNITCOUNT().

For more information on these parameters, refer to the ***TCPaccess Customization Guide.***

SITE Command Parameters

The Server FTP **SITE** command has parameters specifically for using FTP to transfer files to magnetic tape. These are:

- AUTOINDEX
- COMPACT
- DSEQ
- LABEL()
- MOUNT()
- PARALLELMOUNT
- PRIVATE
- TAPE

These parameters are available for using FTP to transfer files to disk, but may have special significance for transferring files to magnetic tape:

- EXPDT
- RETPD
- UCNT
- VCNT
- VOL(volser, volser)
- VSEQ

FTP Server Commands **APPE** and **RESTART** are not currently supported for the FTP to Tape facility.

For more information on these **SITE** parameters, read "SITE" on page 5-15.

Using FTP to Write to Magnetic Tape

Cataloging Data Sets

FTP attempts to catalog all data sets created on tape. If the data set name matches an existing name, the transfer will occur but the catalog will not be updated. This may create a problem if a retrieve is issued for the tape version. Therefore, it is recommended that all data sets—disk or tape—have unique names. Use the **DELETE** command to uncatalog tape data sets.

Cataloged tape data sets are assumed to exist on standard label tapes. Use the **SITE** command with LABEL and DSEQ parameters when retrieving a cataloged tape data set if the tape does not have standard labels. It is recommended that you use standard label tapes whenever possible.

Writing Multiple Data Sets to Tape

Users who want to write several data sets to tape in one FTP session should be aware that each file to be transferred will generate a mount request, but it may be for a different tape unit. Dynamic allocation does not support RETAIN or UNIT AFFINITY. A workaround would be to arrange with Operations to hard mount a tape and then reference the unit in the **SITE** command.

Submitting Mount Requests

The mount request is issued by TCPaccess, even if running client FTP in batch. This means that Operations should not terminate the mount request by cancelling the job requesting the mount. A WTOR message, ACC511A, is issued along with the mount request. A reply of NO to this message cancels the request.

Preventing Timeouts on Data Transfers

FTP will timeout a data transfer request if the remote does not complete the data connection in a certain time. If the remote is another MVS system using tapes (or recalling a data set) it will require a tape mount on the remote system before it can complete the data connection. For this reason, FTP will use the longer of MOUNT or HSM times, or 30 minutes if neither tape support nor HSM support is configured.

Using Tape Data Sets on Remote Hosts

The problem of a remote system using tape data sets should also be considered when configuring DATAIDLE time. If a remote is reading a multivolume , for example, it might have to stop the data transfer between volumes while the next tape is mounted. The DATAIDLE time could expire while this is happening.

Server FTP Commands

"Table 5-1" on page 5-7 lists the Server FTP commands. The TCPaccess Server FTP program supports most of the FTP commands defined in the FTP specification, ***RFC 959, File Transfer Protocol (FTP)***. Server FTP commands are accepted by the local FTP server when submitted by a remote client. The August 2, 2000 FTP server will reply or respond to the FTP commands listed here:

Table 5-1 Server FTP Commands Supported by the TCPaccess Server

Command	Function
ABOR	Abort
ACCT	Account
ALLO	Allocate
APPE	Append (with create)
CDUP	Change to parent directory
CWD	Change working directory
DELE	Delete
HELP	Help
LIST	List
MKD	Make directory
MODE	Transfer mode
NLST	Name list
NOOP	No operation
PASS	Password
PASV	Passive
PORT	Data port
PWD	Print working directory
QUIT	Logout
REIN	Re-initialize
REST	Restart
RETR	Retrieve
RMD	Remove directory
RNFR	Rename from
RNTO	Rename to
SITE	Site parameters
STAT	Status
STOR	Store
STRU	File structure
TYPE	Representation type

Command	Function
USER	User name

Not all of the commands listed in “Table 5-1” on page 5-7 are documented in this publication. Only those which have been enhanced for TCPaccess are included. The others in this list conform to their descriptions in ***RFC 959***. “Table 5-2” on page 5-7 lists commands not supported by the TCPaccess FTP server.

Table 5-2 Commands Not Supported by the TCPaccess Server

Command	Description
SMNT	Structure mount
STOU	Store unique
SYST	System

The supported FTP commands are described in detail throughout the remainder of this chapter. The **HELP** command also provides information about the Server FTP. Read “HELP” on page 5-10 for guidelines on using the **HELP** command.

ALLO

The **ALLO** command allocates a specified amount of disk space for a subsequent **STOR** or **APPE** command. When an **ALLO** command specifies an upper limit on the size of the file as stored in the MVS file system, a **STOR** that starts successfully is guaranteed not to fail because of disk space.

Syntax Description

These are the allowable variations of the **ALLO** command syntax:

ALLO R *logical_record_length*	This form is not included in the FTP specification but is a useful extension allowed by the Server FTP. A comma can replace **R**.
ALLO *byte_count*	
ALLO *byte_count* **R** *logical_record_length*	A comma can replace **R**.

If an **ALLO** command is sent, the subsequent **STOR** or **APPE** commands operate with these MVS **SPACE** parameters in effect:

- T = floor (byte_count / track_length) (where T is the number of disk tracks needed)
- S = max(1, floor(T/5))
- SPACE = (TRK,(S+T,S),RLSE) for STOR or SPACE=(TRK,(,S),RLSE) for APPE

Default

- The default, if no **ALLO** is given, is **(5,3)** tracks, unless this default has been changed by your TCPaccess site administrator.

Usage Guidelines

- You can set the space parameters for creating a disk data set either explicitly with a **SITE SPACE(..)** command or implicitly with an **ALLO** *integer* command. If both commands are given, the **SITE** command **SPACE** parameter takes precedence. If the **ALLO** *integer* is given after a **SITE SPACE(..)** command, the reply is “200 NOTE: Ignored, overridden by site space.”
- **ALLO R** sets the LRECL value for a new data set. Once an **ALLO R** value has been set, a file received with a record longer than this limit is truncated rather than folded.
- An **ALLO R** value makes sense only with a record-structured file. If the Server receives an **ALLO R** command when **STRU F** (file-structure) is specified, the **ALLO** command fails and returns the reply: “503 Command conflicts with previous commands.”

HELP

The **HELP** command gives you introductory and reference information on the Server FTP. Output from the **HELP** command is delivered to you by the control connection; the output can be terminated by the Telnet Break facilities. Read "Telnet Break" on page 5-52.

This **HELP** syntax allows you to request **HELP** for a *command_name* or *section_title* option, but not with both. When no option is specified, general help information is given.

HELP [*command_name* | *section_title*]

Syntax Description

command_name	Command for which help is being requested; valid command-name strings are **ALLO**, **HELP**, **REST**, **SITE**, **STAT**, **STRU**, and **TYPE**.
section_title	Title of a reference source of introductory or reference information provided through the control connection.

Default

Use the **DEFAULT** string to request information on the default data set attributes (DCB parameters) created or used by the Server FTP.

Usage Guidelines

Each MVS site supporting a Server FTP can provide additional help information beyond what is shown here. Valid *section_title* strings for TCPaccess are:

- AECF | AECR | AENF | AENR | AETF | ILF | ILR

  Use these parameters to request help about Server FTP operation with various TYPE and STRU settings. "Table 5-3" on page 5-10 shows the TYPE and STRU settings corresponding to each parameter:

Table 5-3 Using the TYPE and STRU Settings

HELP Parameters	Type	Stru
AECF	**AC** or **EC**	**STRU F**
AECR	**AC** or **EC**	**STRU R**
AENF	**AN** or **EN**	**STRU F**
AENR	**AN** or **EN**	**STRU R**
AETF	**AT** or **ET**	**STRU F**
ILF	**I** or **L**	**STRU F**
ILR	**I** or **L**	**STRU R**

- INTRO

  Use the INTRO string to request an introduction to the use of the FTP Server.

- NEWS

  Use the NEW**S** string to request help on accessing TCPaccess news.

- PATH

  Use the PATH string to request information on MVS path names (data set names, member names, and volumes) and their relationship to the Server FTP.

- SPACE

  Use the SPACE string to request information on MVS space allocation in relationship to the Server FTP.

MKD

The **MKD** (MaKe Directory) command creates a partitioned data set (PDS).

MKD *pathname*

Syntax Description

pathname	Path of the PDS to be created.

Default

The special **GAT TYPE(LIBRARY)** statement (if present) overrides defaults for the **MKD** command.

Usage Guidelines

- The pathname can be either a fully or a partially qualified data set name.
- These are the possible PDS file attributes:
  - TCPaccess space allocation defaults are SPACE(5,3) DIR(5);
  - The DEFGAT initialization statement can provide installation defaults
  - Any **SITE** commands entered override any of the above

Example

```
ftp> pwd
257 "'MVS.'" is current prefix
ftp> mkd mkd.pds
257-"'MVS.MKD.PDS'" partitioned dataset created with attributes:
Volser ICSPK1  Unit SYSALLDA  Dsorg PO  Recfm FB  Lrecl 80
Blksize 6160  Space 5 15 Tracks  Rlse  Dir 46
257
ftp> mkd 'mvs.help.pds'
521 "'MVS.HELP.PDS'" data set already exists.
```

Note: Read the *TCPaccess Customization Guide* for information about the **GAT** statements.

REST

The **REST** (restart) command specifies that the data transfer command that follows immediately is to restart at a specified intermediate point in the file.

REST *restart_marker*

Syntax Description

restart_marker	Marker from which the restart is to begin.

Default

The default interval is every 500,000 data bytes.

Usage Guidelines

- After a **REST** command, **STOR** and **APPE** have identical meanings (**APPE** is taken to mean **STOR**).
- Data transfer must be in MODE B (block mode). The Server can send and accept restart markers in either **STRU F** or **STRU R**.
- A file retrieved from the Server FTP includes restart markers at a specific interval. The **SITE** command RESTART option can change this interval or suppress restart markers entirely. When the count of bytes read from the disk since the last marker reaches the specified interval, a marker is sent at the next end of a complete logical record or segment of a spanned record.

Restart markers

"Table 5-4" on page 5-13 sent by the Server FTP consist of twelve characters that are the ASCII representation of six bytes in the format *VTTRBB.*

Table 5-4 Restart Markers Sent by the Server FTP

Restart Marker	Description
V	Volume sequence number
TTR	Standard MVS disk block address (referred to in IBM publications as a relative track and record address)
BB	A byte offset within a TTR block

RMD

The **RMD** (ReMove Directory) command deletes an empty PDS. It will not delete a PDS that contains members. To delete a sequential file or a PDS containing members, use **DELE**.

RMD *path_name*

Syntax Description

path_name	Name of the PDS to be deleted.

SITE

The **SITE** command supplies host-dependent parameters to the Server FTP, for MVS data management controls, for special FTP controls, and for generic data set attributes.

SITE *parameter* `[ , ]...`

Syntax Description

Available parameters include the following:

```
ATtr (gat_name)
AUTOIndex
AUTOMount | NOAUTOMount
AUTORecall | NOAUTORecall
BLKsize(max_physical_block)
BLocks
BLOCKSIze(blocksize)
CArds | SOurce | FOrtran | OBJect | LOadlib | PRINt
CD | NOCD
CHarset(table_name)
CHKptint(checkpoint_interval)
COmpact | NOCOmpact
CONDdisp(CATLG|DELETE)
CYlinder
DAclass(sms_data_class)
DATAClas(data_class_name)
DATASetmode
DBcsset(table_name)
DCBdsn(data_set_name)
DCLose(data_port_close_time)
DEVNull
DIDle(data_port_idle_time)
DIrectory(blocks)
DIRECTORYMODE
DOpen(data_port_open_time)
DSeq(number)
DUMMy
Expdt(expiration_date) | RETPD(retention_period)
FILEtype(SEQ|JES|VTOC)
FOrtran | CArds | SOurce | OBJect | LOadlib | PRINt
FULLtrk | Halftrk | VBs | VS
Halftrk | FULLtrk | VBs | VS
HFS | MVS
IBuf(numbuf bufsize)
IRBlksize(max_physical_block)
IRLrecl(logical_record_length)
IRRecfm(record_format)
ISPFEnq | NOISPFEnq
ISPFRes | NOISPFRes
JESLrecl(record_length)
JESRecfm(record_format)
LAbel(type)
```

```
LIne(logical_record_length)
LIStfmt(OLD | IBM | SHORT)
LKEDres | NOLKedres
LOadlib | CArds | SOurce | FOrtran | OBJect | PRINt
LRecl(logical_record_length)
MAnagementclas(management_class_name)
MGmtclass(sms_management_class)
MIGratevol(migration_volume_serial)
MOunt(time)
NCP(number_of_DASD_buffers)
NDab(number1 number2)
NLstcase(UPPER | LOWER)
OBJect
OBUf(numbuf bufsize)
OVerwrite
PAD(char, char)
PARallelmount
PDse | NOPDse
PErsist | NOPErsist
POp
PREmount
PRImary
PRINt | LOadlib | CArds | SOurce | FOrtran | OBJect
PRIVate
PUsh | POp
Qdisk(volume_serial_mask)
RDw | NORDw
RECAll(integer) | NORECall
RECFm( F | FB | FBS | FBA | FBSA | V | VB | VS | VBA | VBS | U )
REPlyfmt(OLD | IBM )
RESEt
RESTart(integer)
RETpd(retention_period) | Expdt(expiration_date)
RLse | NORLse
SECondary
SOurce | CArds | FOrtran | OBJect | LOadlib | PRInt
SPace(primary_allocation, secondary_allocation)
STClass(sms_storage_class_name)
STORclas(storage_class_name)
STRip | NOSTRip
SUbmit | NOSUbmit
TABs(integer)
TAPE
TErse
TRACKs
TRANopt(char_translation_mode)
UCNt(unit_count)
UMASK (000)
Unit(unit_name)
VCNt(volume_count)
VErbose
VOlume(volume_name, volume_name, ...)
VSEQ(volume_sequence_number)
VBs | VS | FUlltrk | Halftrk
```

```
VS | VBs | FUlltrk | Halftrk
WRAPrecord | NOWRAPrecord
```

Site Command Parameters

These parameters are described in “Table 5-5” on page 5-17.

Table 5-5 SITE Command Parameters

Parameter	Description
ATTR(*gat_name*)	Specifies any entry in the Generic Attributes table. Read “Generic Attribute Names” on page 5-35 for details. This command is not supported for the FTP to Tape Facility.
AUTOINDEX	Requests that the data set sequence number be increased by one for each subsequent file transfer.
AUTOMOUNT | NOAUTOMOUNT	AUTOMOUNT is an alias for MOUNT. NOAUTOMOUNT is an alias for NOMOUNT.
AUTORECALL | NOAUTORECALL	AUTORECALL is an alias for RECALL. NOAUTORECALL is an alias for NORECALL.
BLKSIZE (*maximum_physical_block_length*) BLOCKSIZE (*maximum_physical_block_length*) LRECL (*logical_record_length*) or LINE (*logical_record_length*) RECFM(*record_format*)	Explicitly sets the DCB or format attributes of a new data set referenced by a STOR or APPE. If the data set is being created, these parameters override the defaults determined by the FTP TYPE or STRU commands. If the data set exists, these parameters must exactly match the corresponding attributes of the data set. BLOCKSIZE is an alias for BLKSIZE. Record formats may be found with the RECFM parameter.
BLOCK	Space allocation is to be in blocks.
CARDS | SOURCE| FORTRAN | OBJECT | LOADLIB | PRINT	Specifies one of the standard Generic Attribute Names supplied with TCPaccess. Read “Generic Attribute Names” on page 5-35 for details.
CD | NOCD	CD enables directory commands (CWD, PWD, CDUP). NOCD disables directory commands. Note The CD | NOCD parameter is not reset when data transfer begins.
CHARSET (*table_name*)	Selects an alternate character set (translation) table for single-byte data transfer ASCII data. This table is validated for single-byte data.
CHKPTINT(*checkpoint_interval*)	Specifies the number of logical records between restart markers.
CONDDISP(CATLG | DELETE)	Specifies the conditional disposition for new data sets created by STOR or APPE when the file transfer fails.
COMPACT	Specifies IDRC compaction for 3480 tapes.
CYLINDER	Space allocation is to be in cylinders.

DACLASS(*sms_data_class*)	Alias for DATACLASS.
DATACLASS(*data_class_name*)	Specifies the SMS data class.
DATASETMODE	Requests the FTP server to display directory output (LIST/NLST) in data set mode. Each data set is listed individually.
DBCSSET(*table_name*)	Selects an alternate character set (translation) table used for double-byte data transfer ASCII data. This table will be validated for double-byte data.
DCBDSN(*data_set_name*)	Specifies a model data set for data set attributes.
DCLOSE(*data_port_close_time*)	Specifies the time, in seconds, FTP will wait to close a data port.
DEVNULL	Requests that the FTP server allocate a dummy (NULLFILE) data set for storing a data set with the STOR command.
DIDLE(*data_port_idle_time*)	Specifies the time, in seconds, FTP will wait on an idle data port. Minimum value = 60 seconds. Maximum value=1439 minutes.
DIR(*blocks*)	Integer number of 256-byte blocks to be reserved for a PDS directory. One block holds from 7 (load module) to 16 (source module) member entries. This parameter is required to create a new PDS with a STOR or APPE command.

DIRECTORYMODE	Requests that the FTP server display directory output (LIST/NLST) in directory mode. Data sets which have the same qualifier at the level immediately below the prefix level are grouped together as “pseudo-directories”.
	Example
	The following example illustrates the use of the DIRECTORY MODE parameter.
	`ftp> quote site directorymode`
	`200 OK, Ready`
	`ftp> dir v*`
	`125 List started OK.`
	`Volume Unit Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname`
	`ICS009 3390 03/19/96 1 1 VS`
	`29389 29393 PS VS`
	`Pseudo Directory`
	`V1`
	`91`
	`Pseudo Directory`
	`V`
	`211`
	`Pseudo Directory`
	`V`
	`311`
	`Pseudo Directory`
	`V`
	`410`
	`250 List completed successfully.`

DOPEN(*data_port_open_time*)	Specifies the time, in seconds, FTP will wait to open a data port.
DSEQ	Specifies data set sequence number.
DUMMY	This is an alias for DEVNULL.
EXPDT | RETPD	Specifies expiration date and retention period. EXPDT(*expiration_date*)—Specifies an expiration date for a new data set, in the format: expiration_date = yyyyddd or yyyy/ddd Note yyyy is a year from 1900 to 2155, ddd is a julian date from 1 to 366. You must include any leading zeroes in the ddd value. RETPD(*retention_period*)—Specifies a retention period for a new data set retention_period is a number of days between 1 and 9999. Note EXPDT and RETPD are mutually exclusive.

FILETYPE(SEQ | JES | VTOC)	Specifies the type of file the FTP server is working with: SEQ—Sequential files. This command is the same as issuing a SITE NOSUBMIT command. JES—JES spool. Data is written to the JES internal reader. FILETYPE=JES is the same as issuing a SITE SUBMIT command. VTOC—DASD VTOC records. Directory commands list statistics from the Volume Table of Contents for DASD volumes. Default: SEQ Note You cannot do a GET of spool files or display the JES spool queue with the DIR command.
FORTRAN | CARDS | SOURCE| OBJECT | LOADLIB | PRINT	See the description for CARDS.
FULLTRK | HALFTRK | VBS | VS	Specifies general attributes for a data set.
HALFTRK | FULLTRK | VBS | VS	See the description for FULLTRK.
HFS | MVS	Specifies the change directory. HFS indicates the change directory should be treated as an HFS filename. MVS indicates the change directory should be treated as an MVS dataset.
IBUF(*numbuf bufsize*)	Specifies in sublist notation the number of network input buffers (numbuf) and the buffer size (*bufsize*) to be used during data transfer. The maximum value for each number is 32767. Note Your TCPaccess system administrator might have set restrictions on use of the IBUF parameter.
IRBLKSIZE(*max_physical_block*) IRLRECL(*logical_record_length*) IRRECFM(*record_format*)	Explicitly set the DCB or format attributes to be used to allocate the internal reader data set when SITE SUBMIT has been entered and a data transfer is performed.
ISPFENQ | NOISPFENQ	Specifies that the ISPF enqueue facility be activated (ISPFENQ) or deactivated (NOISPFENQ). Default: NOISPFENQ
ISPFRES | NOISPFRES	Enables (ISPFRES) or disables (NOISPFRES) the RESERVE logic for the SPFEDIT ENQ, if the volume on which the PDS resides is shared by Multiple Systems (UCB shared bit ON).This assures data integrity while the PDS you are accessing is being simultaneously accessed by an ISPF user from another system. Default: NOISPFRES
JESLRECL(*record_length*)	Alias for IRLRECL.
JESRECFM(*record_format*)	Alias for IRRECFM.
LABEL	Specifies label type. Label options supported are: SL, NL, BLP, LTM, AL. See LABEL parameter description in the description of the GAT statement in the TCPaccess Customization Guide.

LKEDRES | NOLKEDRES	Enables (LKEDRES) or disables (NOLKEDRES) the RESERVE logic for the SYSIEWLP ENQ, if the volume on which the PDS resides is shared by Multiple Systems (UCB shared bit ON). This assures data integrity while the PDS you are accessing is being simultaneously accessed by the linkage editor from another system. Default: NOLKEDRES
LINE(*logical_record_length*)	See BLKSIZE.
LISTFMT(OLD | IBM | SHORT)	Specifies whether output from the data set LIST command will be in the old TCPaccess format, in IBM-standard format, or in a shortened IBM-compatible format. The short format leaves out data set extents and tracks allocated, but improves LIST response time. Default: SHORT Note Certain PC-based client FTP packages expect the LIST output from a host configured as OS/MVS to be in standard IBM format. The LIST parameter is not reset when data transfer begins.
LOADLIB | CARDS | SOURCE| FORTRAN | OBJECT | PRINT	See the description for CARDS.
LRECL(*logical_record_length*)	See BLKSIZE.
MANAGEMENTCLASS (*management_class_name*)	Specifies the SMS management class.
MGMTCLASS (*sms_management_class*)	Alias for MANAGEMENTCLASS.
MIGRATEVOL (*migration_volume_serial*)	The volume serial number of migrated data sets.
MOUNT(*time*)	Enables tape support ability for this file transfer. The time value specifies the maximum wait time in minutes for the tape mount to complete. If the time expires, the request is aborted. MOUNT without a time value specified will use the default system wait time.
NCP(*number_of_DASD_buffers*)	Alias of NDAB.
NDAB(*number1 number2*)	Specifies the number of DASD buffers used by FTP for reading or writing disk data sets. The maximum value is 99. Default: 4 Note Your TCPaccess administrator might have set restrictions on the use of the NDAB parameter.

NLSTCASE(UPPER | LOWER)	Specifies whether the output from an NLST command will be upper- or lowercase. If LOWER is specified and the data set or member list is part of the current directory, the names will be returned in lowercase. Note NLSTCASE(LOWER) is supplied to facilitate MGET functions from FTP clients on systems that use lowercase file names. Note This parameter is not reset when data transfer begins.
OBJECT | LOADLIB | CARDS | SOURCE| FORTRAN | PRINT	See the description for CARDS.
OBUF(*numbuf bufsize*)	Specifies, in sublist notation, the number of network input buffers (numbuf) and the buffer size (bufsize) to be used during data transfer. The maximum value for each number is 32767. Note Your TCPaccess system administrator might have set restrictions on use of the OBUF parameter.
OVERWRITE | NOOVERWRITE	This parameter is a toggle. OVERWRITE requests that the FTP server overwrite an existing data set when transferring files if a data set of the same name already exists on the target server. This parameter is necessary when the SITEOVERWRITE configuration parameter is in effect. Note This parameter is reset after each transfer, regardless of the PERSIST option in effect.
PAD (*pad_code*)	Overrides the default characters that pad network records or lines to fixed-length logical records when data is stored (via STOR or APPE) or deleted from fixed-length logical records when data is retrieved (via RETR). The defaults are blanks for character types and zeros for binary types. These are the pad-codes: Z – Pad with zeros O – Pad with ones B – Pad with blanks
PARALLELMOUNT	Specifies parallel mounting (mutually exclusive with UCNT).
PDSE | NOPDSE	Allocates PDSEs instead of PDSs, or vice versa.
PERSIST | NOPERSIST	Specifies whether SITE parameters will be reset following data transfer. If PERSIST is used, all SITE parameters remain in effect until explicitly changed via subsequent SITE commands, or reset with SITE RESET. If NOPERSIST is used, all SITE parameters are reset after each data transfer. Default: NOPERSIST
PRIMARY(*primary_allocation*)	Specifies the primary space allocation for data sets.
PRINT | LOADLIB | CARDS | SOURCE| FORTRAN	See the description for CARDS.
PRIVATE	Requests a private volume.

PUSH | POP	PUSH causes the server FTP to save the current settings of parameters entered with previous SITE commands; POP restores those parameters. Note PUSH and POP can be nested. Each PUSH adds an entry to a push-down stack. Each POP pulls the last entry from the stack. When there are no entries in the stack, a POP will result in an error reply.

QDISK(*volume_serial_mask*)	Requests the FTP server to provide volume information for the volume(s) specified in the *volume_serial_mask*.
	Example
	`ftp> quote site q=ics001`
	`200- % Free Free Largest Free`
	`200- Volume Unit Free Cyls Trks Cyls-Trks Exts Address Use Attr`
	`200- ICS001 3390 1 23 102 15 14 23 420 Storage`
	`200 Site command was accepted`

RDW | NORDW	Specifies whether RDWs (Record Descriptor Words) will be sent as data for RECFM=VB and RECFM=VBS files. If RDW is selected, the RDW will be sent for binary, ASCII, or EBCDIC transfers. Default: NORDW Note RDW will not be translated for ASCII file transfers. This parameter is ignored for data sets with carriage control (RECFM=VBA or RECFM=VBM).
RECALL(*integer*) | NORECALL	RECALL—Enables Hierarchical Storage Manager (HSM) recall ability for this file transfer. The integer value specifies the maximum wait time in minutes for HSM to complete the recall of the migrated file. If the time expires, the request is aborted. RECALL without an integer value specified enables HSM with the default system wait time. NORECALL—Disables HSM recall ability for this file transfer.
RECFM(F | FB | FBS | FBA | FBSA | V | VB | VS | VBA | VBS | U))	Record format.
REPLYFMT (OLD | IBM)	Specifies whether reply output, when in FTP/JES mode, will be in IBM reply numbers and text as opposed to the TCPaccess format. Default: OLD
RESET	Resets all previous SITE commands. Can be used if SITE PERSIST is specified.

RESTART (*integer*)	When a file is RETRieved in block mode, the FTP Server includes a Restart Marker in the data stream every integer data bytes. To suppress these markers entirely, specify RESTART(0). The default is RESTART(500,000). Read "REST" on page 5-13 for information about Restart processing. This command is not supported for the FTP to Tape Facility.
RETPD | EXPDT	See the description for EXPDT.
RLSE | NORLSE	RLSE—Cancels a previous SITE NORLSE. This is needed occasionally to prevent building up too many extents when many APPE operations to the same data set are performed. NORLSE—Specifies that unused disk space not be released following a STOR or APPE. Default = RLSE
SECONDARY (*secondary_allocation*)	Specifies the secondary space allocation for data sets.
SOURCE | CARDS | FORTRAN | OBJECT | LOADLIB | PRINT	See the description for CARDS.
SPACE(*primary_allocation*, *secondary_allocation*)	Specifies primary and secondary disk space allocation (the default allocation is in tracks). Only required for STOR or APPE commands.
STCLASS (*sms_storage_class_name*)	Alias for STORCLASS.
STORCLAS(*storage_class_name*)	Specifies the SMS storage class.
STRIP | NOSTRIP	Specifies whether pad characters will be stripped from fixed-length logical records when data is retrieved (using RETR).
SUBMIT | NOSUBMIT	SUBMIT—Specifies the MVS FTP Server to send the output of a file transfer to a JES internal reader for execution. NOSUBMIT—Cancels a previous SITE SUBMIT.
TABS (*integer*)	Specifies the tab stop interval to be used in receiving the next file. The default is 8, and the limit is 25. TABS(0) translates tab characters (for example, ASCII x'09' is translated to EBCDIC x'05'); TABS(1) replaces each tab character with a blank.
TAPE	Specifies that attributes are to be taken from the GAT TYPE(TAPE) entry.
TERSE	Requests the FTP server to issue single-line 150 and 226 replies.
TRACKS	Requets space allocation in tracks.

TRANOPT (*char_translation_mode*)	Defines the character translation mode. These are the choices: CHAR—Defines character translation mode as single-byte. DBCS—Defines character translation mode as double-byte. MIX—Defines character translation mode as single-byte and/or double-byte.
UCNT(*unit_count*)	Specifies the maximum number of generic units an output data set can require. A value of 1 to 59 can be entered.
UNIT(*unit_name*)	Specifies a generic unit for creating a new data set.
UMASK (000)	Allows you to specify a 3 character octal number to set file access defaults. You must specify this at the beginning of the FTP session.
VBS | VS | FULLTRK | HALFTRK	See the description for FULLTRK.
VCNT(*volume_count*)	Specifies the maximum number of volumes an output data set can require. A value of 1 to 255 can be entered.
VERBOSE	Requests the FTP server to issue multi-line 150 and 226 replies, showing data set attributes and transfer statistics.
VOLUME(*volume_name*, *volume_name*, ...)	Specifies an explicit volume(s) for creating a new data set or referencing an old uncataloged data set. Normally not required. A total of 255 can be entered. If VOLUME is entered with no volume_name, all VOLUME information is reset.
VS | VBS | FULLTRK | HALFTRK	See the description for FULLTRK.
VSEQ(*volume_sequence_number*)	Specifies that processing should begin at a requested volume within a multi-volume data set. A value of 1 to 255 can be entered.
WRAPRECORD | NOWRAPRECORD	Network records which exceed LRECL will be wrapped to the next record when receiving data.

Usage Guidelines

- **SITE** command keywords are evaluated in left to right order. Successive **SITE** commands are cumulative. A later **SITE** command can add to or change attributes established by an earlier **SITE** command.
- **SITE** command parameters can be entered in either PL/I or BAL formats. The examples for FTP3 are shown in PL/I format. Any **SITE** parameter that takes a keyword can be entered in either format.
- If NOPERSIST is used, all **SITE** parameters are reset after each data transfer. If PERSIST is used, all **SITE** parameters remain in effect until explicitly changed via subsequent **SITE** commands, or reset with **SITE RESET**.
- When a sub-parameter takes a list and more than one value is contained in the list, the list of values must be enclosed in parenthesis. Even if you choose to enter the command in BAL format, you must use parentheses around the list.
- The **SITE** command verb is followed by a list of keyword parameters. Each keyword may normally be abbreviated to the minimum-length unambiguous string, as in TSO. (In the preceding list, the minimum abbreviation for each parameter is shown in upper-case.)
- **SITE** parameters must be separated by a comma or a blank.
- A single FTP command is limited to 80 characters. In the (unlikely) event that a **SITE** command exceeds

80 characters, it can be broken into two or more successive **SITE** commands.

- If an error is found in parsing a **SITE** parameter, an error message is issued indicating the bad parameter, and the FTP server continues with the next parameter.
- HFS, MVS, and UMASK are persistent for the session. If you do not specify HFS or MVS on the SITE command, then Server FTP uses the change directory to determine the format.

Examples

The following example illustrates the BAL format:

SITE ATTR=*gat_name*

The following example illustrates the PL/I format:

SITE ATtr (*gat_name***)**

The following example illustrates the obligatory use of parentheses when the command is entered in BAL format:

SITE VOLUME=(*vol1***,** *vol2***,** *vol3***)**

STAT

The **STAT** command provides partial or complete status of the Server FTP.

STAT

Syntax Description

selector	A string containing any subset of the letters **F** | **A** | **P** | **T** | **I** • **F**—FTP parameters, such as **MODE** and **TYPE**. • **A**—Access control, such as user ID and account. • **P**—Path data, such as data set name and attributes. • **T**—Data transfer status (such as number of bytes, records transmitted); null if no transfer is in progress • **I**—Internal control blocks of Server FTP.
item_numbers	One or more positive integers separated by commas.
path_name	A valid MVS directory identifier optionally enclosed in quotes.

Usage Guidelines

- The commands **STAT** or **STAT ?** mean **STAT ? FAPT**.
- The command **STAT ?** *item_numbers* gives only the specified item of the full **STAT** display. This is useful in debugging since the full display can be lengthy.
- The command **STAT** *path_name* (where *path_name* is a valid MVS data set prefix specified in the form *myuid*.) gives catalog information on a specific group of data sets. **STAT** *path_name* gives the same information as **LIST** but sends it over the Telnet control connection instead of a data transfer connection.
- The command **STAT *** gives the catalog list for the default (logged-in) directory.

Example

STAT ?

STAT ? *selector*

STAT ? *item_numbers*

STAT *path_name*

Data Set Attributes

This section describes the attributes of data sets that can be retrieved (read) or stored (written) by the Server FTP.

Units and Volumes

The Server FTP can write and retrieve only disk data sets stored on permanent-resident disk volumes.

Data Set Names

The Server FTP rules for naming and accessing data sets described here are the same as those for TSO.

User disk data sets generally have names of the form:

```
defprfx.name1[.name2[.name3 .. [.namen]]]
```

Syntax Description

defprfx	That portion of the data set name that is the defined default prefix of the installation.
namex	Data set name indexes, made up of 1 to 8 alphanumeric characters, the first of which is alphabetic .

Note: The total length of the data set name, including the periods between indexes, cannot exceed 44 characters.

Data set naming conventions can vary between MVS sites. Consult personnel at your site to learn naming conventions.

When the data set name is enclosed in single quotes, it is a fully qualified data set name. When the data set name is not enclosed in quotes, it is partially qualified. Under this Server FTP (and TSO), you normally specify data set names in a partially qualified fashion, allowing the system to prefix the installation's default prefix to the data set name.

Server FTP uses one of the three possible default prefixes:

- User ID
- none
- character strings.

If User ID is defined at your installation, then server FTP uses the User ID provided by the **USER** FTP command as the *defprfx* for prefixing. If no prefixing is defined, the data set is fully qualified and quotes are not required. If character string is defined, the installation selected a common qualifier for all data sets. Individuals can select their own data set prefix with the Change Working Directory (**CWD**) or Change to Parent Directory (**CDUP**) FTP commands.

Example

These are some examples of how to set your prefix. The first method shows the **cd** and **cdup** User commands; the second shows sending the **cwd** and **cdup** commands to the server using **quote**.

```
ftp > pwd
257 "'MYID.'" is current prefix.
ftp > cd level1
250 "'MYID.LEVEL1.'" is current prefix.
ftp > cdup
200 "'MYID.'" is current prefit
ftp > cd level2
250 "'MYID.LEVEL2.'" is current prefix.
ftp > cd 'newid'
250 "'NEWID.'" is current prefix.
ftp > quote cwd 'nextid'
250 "'NEXTID.'" is current prefix.
ftp > quote cdup
200 No prefix defined.
```

The Server FTP supports both simple sequential and partitioned data sets. A PDS contains an internal directory to a set of sub-data sets called members. All members of a PDS share the same data set name (*dsname*) and attributes. The fully qualified name of a PDS member is:

`defprfx.`***name1***`[.`*`name2`*`[.`*`name3`*` .. [.`*`namen`*`]]](`*`member_name`*`)`

The *member_name* field has the same syntax as *name1* through *namen*.

The Server FTP also supports Generation Data Group (GDG) data sets. A GDG data set has a similar format to a PDS but the member name takes the form 0, **+***n*, or **-***n*, where *n* is the relative generation number.

FTP Path Name Syntax

The data transfer commands **STOR**, **APPE**, **RETR**, **DELE**, **RNFR**, and **RNTO** have *path_name* as a parameter. With this Server FTP, *path_name* specifies a data set or PDS member in one of these forms:

- *name1[.name2[.name3 ... [.namen]]]*
- *name1[.name2[.name3 ... [.namen]]](member_name)*
- *name1[.name2[.name3 ... [.namen]]](GDG_number)*
- *'name1[.name2[.name3 ... [.namen]]]'*
- *'name1[.name2[.name3 ... [.namen]]](member_name)'*
- *'name1[.name2[.name3 ... [.namen]]](GDG_number)'*

The first three forms are partially qualified data set names. An installation defined default prefix is prefixed to names in these forms. Typically, you use one of these forms to refer to your own data sets. The last three forms are fully qualified and are used as is by the FTP Server. The second and fourth forms are used for PDS libraries and the third and sixth forms are used for GDGs.

In addition, a *member_name* can be entered alone when the current directory is a partitioned data set. Read "VTOC and Catalog" on page 5-32.

Note: The length of the 1- to 8- character member name or the GDG number and the enclosing parentheses are not included in the 44-character limit.

Using Wildcard Characters in FTP

The server commands **STAT**, **LIST**, and **NLST** accept a data set or member name mask as a parameter. The output from the command lists all data sets or members that match the mask criteria.

These are the rules for masking:

- An asterisk (*) represents zero or more consecutive characters
- A percent sign (%) represents a single character

Note: If a member name or member name mask is included, the data set name must not include a mask.

Example

```
ftp> ls v*.obj
200 OK, Ready
125 Transfer started
V111.OBJ
V20.OBJ
V201.OBJ
226 Transfer complete
ftp> ls v2%.obj
200 OK, Ready
125 Transfer started
V20.OBJ
226 Transfer complete
ftp> cd v20.obj
250 "'MVS.V20.OBJ'" partitioned data set is current directory
ftp> ls ftps*
200 OK, Ready
125 Transfer started
FTPS
FTPSFTDR
226 Transfer complete
ftp> ls v*.obj(*)
200 OK, Ready
501 Wildcard characters are not permitted within a Partitioned dataset name
```

Partitioned Data Sets

When a **CWD** or **CDUP** command causes the prefix to match the data set name of a cataloged PDS, the PDS becomes the working directory. Subsequent data transfer commands **STOR**, **RETR**, **DELE**, **RNFR**, and **RNTO**, as well as the **LIST**, **NLST**, and **STAT** commands, will treat unquoted path names as PDS member names. In addition, the **LIST** and **NLST** commands without a path name cause a list of members for the PDS directory to be output.

Examples

```
ftp> pwd
257 "'MVS.'" is the current prefix
ftp> cd help.pds
250 "'MVS.HELP.PDS'" partitioned data set is current directory
ftp> dir
200 OK, Ready
125 Transfer started
  Name     VV.MM  Created    Changed     Size  Init  Mod  Id
FTPDEFAU   04.00  12/08/93   12/09/93    6:16   98   98  0 MVS
FTPINTRO   01.00  12/09/93   12/09/93    6:18  134  134  0 MVS
FTPNEWS    01:00  12/09/93   12/09/93   10:30   25   25  0 MVS
GREETING   01:00  12/08/93   12/09/93    6:19   12   12  0 MVS
226 Transfer complete
ftp> get ftpnews
200 OK, Ready
150-Dataset open with attributes:
Type A N Tabs 8 Stru F Mode S Path MVS.HELP.PDS(FTPNEWS)
Volser MVSVOL Unit SYSALLDA Dsorg PO Recfm FB Lrecl 80
Blksize 3120 Rlse
150
226 Transfer complete
```

In addition, the **LIST**, **NLST**, and **STAT** commands will accept member name masks as path names. The member name mask can be entered either alone (if the current directory is a PDS) or as part of a fully qualified PDS data set name.

```
ftp> ls 'mvs.help.pds(ftp*)'
200 OK, Ready
125 Transfer started
'MVS.HELP.PDS(FTPDEFAU)'
'MVS.HELP.PDS(FTPINTRO)'
'MVS.HELP.PDS(FTPNEWS)'
226 Transfer complete
```

If you want to treat a PDS data set path name as a prefix, enclose the fully-qualified name in quotes and append a period (.) at the end.

```
ftp> cd 'mvs.help.pds.'
250 "'MVS.HELP.PDS.'" is current prefix
```

VTOC and Catalog

Under MVS, each disk volume contains a file directory of the data sets on that volume called the volume table of contents (VTOC). Hence, any disk data set can be located via the logical path name: *volume,dsname*. The Server FTP lets you specify the volume name in the **SITE** command. Read "SITE" on page 5-15 for details.

MVS has a central file directory, called the catalog, that provides the volume name as a function of the *dsname*. TSO generally requires that all user data sets except scratch files be cataloged (i.e., listed in a system catalog). A cataloged data set name is unique on the system while an uncataloged data set name need be unique only on the disk volume where it resides. You need to give only the *dsname* (and *member_name* for a PDS), not the volume, to locate an existing cataloged data set.

This Server FTP generally does not change the catalog status of a data set operated on by a **STOR**, **APPE**, **RETR**, or rename operation. A data set created by an FTP **STOR**, **APPE**, or **RNTO** operation is cataloged. If the *dsname* conflicts with an existing cataloged data set, the operation is refused in the case of **RNTO**.

When the Server FTP performs an **APPE** or **RETR** operation on an existing uncataloged data set (using **SITE VOLUME**), it tries to catalog it. If the attempt fails because another data set with the same name is being cataloged or because the *dsname* has the wrong tree structure, a warning message is sent but the operation proceeds.

MVS Space Allocation

MVS allocates disk space to a data set in variable-sized pieces called extents. Each extent is a contiguous set of disk tracks. The byte capacity of a disk track depends on the disk model. For example, A 3380 holds a maximum of 47476 bytes per track. A data set used by the Server FTP is limited to a maximum of 16 extents. Therefore, if the disk space is fragmented, you might run out of extents before running out of total space. In such a situation, choice of appropriate space parameters is important.

Space allocation may have two parameters to the operating system: a primary space quantity, and a secondary space quantity. When an FTP **STOR** operation is requested, FTP first tries to allocate the primary quantity. If it succeeds, data transfer starts. Each time that space is exhausted during the **STOR** (or **APPE**) operation, FTP tries to allocate the secondary quantity. The **STOR** (or **APPE**) fails at that point if it cannot satisfy the request. Each allocation, whether primary or secondary, takes place as follows:

- MVS tries to find a contiguous area (extent) to satisfy the request using the best-fit algorithm
- If no single area is large enough, it uses the fewest extents possible (up to five) to satisfy the request

This allocation process continues until the total space or the extent limit is exhausted.

The default space parameters for FTP **STOR** are (primary, secondary) = (5,3) tracks. The maximum space that can be allocated with these parameters depends on the degree of fragmentation of the volume (assuming that enough total space exists). It ranges from 50 tracks down to 14 tracks (if the largest contiguous area is only 1 track and extents are exhausted first).

These methods obtain more space or a larger file:

- Send an **ALLO** command before the **STOR**.

The parameter to **ALLO** is a file size in (network) bytes. The Server FTP converts that value to disk tracks and uses 1.2 times that value as the initial space allocation. Therefore, if the data transfer starts, you know the initial space allocation was successful and you cannot run out of space specified by the **ALLO**. The 1.2

factor is intended to take care of inter-record gaps. The secondary space quantity is 0.2 times the **ALLO** value.

- Specify explicit **SPACE** parameter on the **SITE** command.

The disk space allocation (**ALLO**) is in terms of the data stored on disk, including padding. A text file normally is padded to 80 byte records, so each line is 80 bytes on disk. The primary and secondary space quantities are recorded in the data set label (**DSCB**) by the operating system. A subsequent **APPE** uses the secondary quantity determined when the data set was created unless you override it with a new **ALLO** or **SITE SPACE** before the **APPE**.

Multivolume Data Sets

If a data set might require more space than is available on a single volume, you can specify that it may reside on multiple volumes using the **SITE VOLUME**, **VCNT**, or **UCNT** parameters. Enough space must exist on the first volume for the primary extent. When the space on the first volume is exhausted, extents will be allocated on the next volume. Up to 16 extents can be allocated on each volume.

FTP will use the greatest of the following numbers to determine how many devices and volumes to allocate to a data set:

- Unit-count specified in the **SITE UCNT** parameter
- Volume-count specified in the **SITE VCNT** parameter
- Number of serial numbers specified in the **SITE VOLUME** parameter

Data Set Organization

FTP supports sequential (**DSORG=PS**) and partitioned (**DSORG=PO**) data sets. Direct access (**DSORG=DA**) data sets can be read sequentially but cannot be written by the Server FTP. ISAM and VSAM data sets are not supported.

Disk Format (DCB) Attributes

A data set under MVS has the DCB attributes listed in “Table 5-6” on page 5-33; the first column shows the **SITE** command keyword parameter(s) to set the corresponding DCB attribute:

Table 5-6 Disk Format Attributes

Keyword	Corresponding data set (DCB) attribute
RECFM	Record Format (**RECFM**)
LRECL or **LINE**	Logical Record Length (**LRECL**)
BLKSIZE or **BLOCK**	Physical Block Length (**BLKSIZE**)

When an existing data set is retrieved, the Server FTP determines the attributes from MVS and uses them to read the disk data set correctly. However, when writing to a data set, either through a **STOR** or **APPE** operation, you may need to be concerned about setting the correct attributes. Whether Server FTP assigns attributes for an existing data set depends on whether the data set is partitioned (PDS) or sequential.

Sequential Data Sets

A **STOR** or an **APPE** to a non-existent data set creates a new data set. The Server FTP assigns the attributes of the new data set. Additionally, the **STOR** operation assigns new attributes to an existing data set.

New attributes are set from default attributes chosen by the Server FTP based on the **TYPE** and **STRU** transfer parameters (read "Default Data Set Attributes" on page 5-34), or from explicit overrides of these defaults by **SITE** parameters.

Partitioned Data Sets

A **STOR** of a member into an existing PDS adds information to the data set (similar to an **APPE**). The attributes of the data set do not change. Any attributes set by the **SITE** command must match those of the existing PDS, or the Server FTP responds with this message:

```
554 SITE LRECL, BLKSIZE or RECFM do not match those of existing data set
```

An **APPE** of a new PDS member is identical to a **STOR**. However, you cannot **APPE** into an existing member because the file system replaces the member. If you attempt to **APPE** into an existing member name, Server FTP responds with this message:

```
504 Not implemented for that parameter, ignored.
```

If a new PDS is being created, either by **STOR** or **APPE**, new data set attributes are assigned by the rules in "Sequential Data Sets" on page 5-34.

Default Data Set Attributes

The Server FTP chooses default data set attributes for **STRU** based on the **TYPE** and chooses **STRU** transfer parameters as shown in "Table 5-7" on page 5-34:

Table 5-7 Default Data Set Attributes

Type	STRU F	STRU R
AN or EN	SOURCE	SOURCE
AT or ET	PRINT	PRINT
AC or EC	PRINT	PRINT
I or L	VS	VBS

Note: In the **STRU R** case, the default data set attributes depend on the size passed by the **ALLO** command. If the **ALLO** command has not been specified or if **ALLO R** is less than 81, the default data set attribute is SOURCE. Otherwise (**ALLO R** is greater than 80), the default data set attribute is VBS, with the LRECL set to **ALLO R** plus 4.

Generic Attribute Names

You can explicitly provide a subset of DCB attributes and DD statement fields on a **SITE** command. Alternatively, the **SITE** command can specify one of the generic attribute names given in "Table 5-8" on page 5-35:

Table 5-8 Generic Attribute Names

Attribute name	RECFM	LRECL	BLKSIZE
FULLTRK	FB	80	**full-track** (see Note **1** following this table)
HALFTRK	FB	80	half-track (see Note **1** following this table)
LOADLIB	U	0	rcmd (see Note **2** following this table)
OBJECT	FB	80	rcmd (see Note **2** following this table)
PRINT	VBA	133	rcmd
SOURCE CARDS FORTRAN	FB	80 (default)	rcmd
VBS	VS	rcmd-4	rcmd
VS	VS	rcmd-4	rcmd

Note: Actual values for half and full track blocking depend on the output disk device type. These generic types create PDS libraries suitable for object modules and load modules.

Each generic attribute name includes values for RECFM, LRECL, and BLKSIZE, and optionally, UNIT, VOLUME, or SPACE, but the attributes set by a generic attribute name can be overridden by other generic or specific attribute keywords. When overriding **SITE** command keywords, the keywords are interpreted in left-to-right order.

In addition, other generic attribute names can be defined by your site. These names can stand for other combinations of RECFM, LRECL, BLKSIZE, UNIT, VOLUME, and SPACE. User-defined names must specify RECFM, LRECL, and BLKSIZE. They can optionally specify UNIT, VOLUME, or SPACE parameters.

The generic types SOURCE, CARDS, FORTRAN, OBJECT, LOADLIB, and PRINT can also carry UNIT, VOLUME, and SPACE parameters as defined by your site. They can be referenced by the **SITE** keywords SOURCE, CARDS, FORTRAN, OBJECT, LOADLIB, and PRINT. Other generic types may also have been defined by your site. They can be referenced by the **SITE** keyword ATTR (type).

In the preceding table, the block size is sometimes specified as rcmd, meaning a recommended value. This means the Server FTP chooses an actual default BLKSIZE that is optimum for the particular device on which the data set is allocated and less than or equal to a recommended size. The recommended size is a site-specific FTP parameter. If your site has not changed this parameter, the recommended size is 6K (6144) bytes. The choice of optimum BLKSIZE <=rcmd depends on RECFM, LRECL, and the disk device type.

Rules for Record Formats

These rules govern Server FTP support of the record format (RECFM) data set (DCB) attribute:

- The Server FTP writes into a disk data set using any of the record formats (RECFM) listed in "Table 5-9" on page 5-36:

Table 5-9 Record Formats

Record format	Description
U, F, V, VS	Unblocked
FB, FBS, VB, VBS	Blocked, nonprint format
FBA, FBSA, VBA	Blocked, print format

- The Server FTP does not support storage or retrieval of data sets with the T (record overflow) or M (machine carriage control) attributes. However, it does support VBM in binary mode.
- These unusual and unblocked print format RECFMs can be read but not written by the FTP Server:

```
FA, FSA, VA, VSA, UA
```

- When the TYPE is I or L (binary data), the data set RECFM cannot specify print format (A). A print data set can be created or retrieved only as text, not as binary data.

Violation of any of these rules results in a 554 illegal recfm error reply.

Data Set Attribute Errors

These conflicting data set attributes create the following error reply and prevent the operation:

```
501 LRECL or BLKSIZE invalid or inconsistent
```

- RECFM ("V.."): BLKSIZE < 4
- RECFM ("VB" or "VBA"): LRECL > BLKSIZE-4 or LRECL < 8
- RECFM ("FB"): LRECL > BLKSIZE or BLKSIZE not an integer multiple of LRECL

In addition, for unblocked RECFMs the LRECL is forced to do the following:

- RECFM ("V"): BLKSIZE-4
- RECFM ("F"): BLKSIZE

Appending to Empty Data Sets

These rules apply to append (**APPE**) operations performed to empty data sets.

- If the data set to be appended to has RECFM=0 or BLKSIZE=0, it is assumed to be empty and is scratched and recreated using the new attributes.
- If the data set is not empty but has LRECL=0 and blocked RECFM, Server FTP uses the new LRECL (the default) or the LRECL from **SITE**.
- Otherwise, Server FTP uses DCB parameters of the existing data set.

JES Internal Reader Support Procedures

These rules apply to the Server FTP **SUBMIT** operation:

- Connect and log on to the host where the JCL resides.
- Connect and log on to your IBM host where TCPaccess is running.
- Use a **QUOTE** type command to send a **SITE SUBMIT** command to the IBM host (such as, **QUOTE SITE SUBMIT**).
- Issue a **PUT**, **GET**, or **SEND** command of the data set on the host where the JCL resides to any name on the IBM host (such as, **PUT IEBPTPCH** *any_name*). The JCL can reside in a physical sequential data set or as a member of a partitioned data set. The new name (*any_name*) on the IBM host is ignored and TCPaccess submits the job to the JES internal reader.

Defaults

The server FTP defaults to RECFM=FB,LRECL=80,BLKSIZE=20000.

The defaults can be overridden by the TCPaccess administrator using the special **GAT TYPE(INTRDR)** entry.

Usage Guidelines

- The DCB attributes used by Server FTP to allocate the JES internal reader can be explicitly set by the user with the server FTP **SITE IRRECFM**, **IRLRECL**, and **IRBLKSIZE** commands.
- Server FTP does a minimum of validity checking for internal reader file attributes. Be sure that the attributes selected are compatible with each other and are appropriate for the local Job Entry Subsystem.

Examples

The following examples show JCL streams that can be submitted using SITE SUBMIT to cause the printing of the data included in the submitted job:

```
//IEBPTPCH JOBCARD
//*
//*     IF THIS JOB STREAM RESIDES ON YOUR REMOTE HOST, YOU CAN
//*     SUBMIT THIS JOB TO THE JES INTERNAL READER OF
//*
//*     YOUR IBM SYSTEM BY ENTERING THE FOLLOWING COMMANDS:
//*
//*     FTP MVS * CONNECT TO THE IBM HOST
//*     USERID/PSW * LOGIN TO THE IBM HOST
//*     QUOTE SITE SUBMIT * NOTIFY SNS/TCP SFTP TO
//*     PUT IEBPTPCH * SUBMIT JCL TO JES INTERNAL READER
//PTPCH       EXEC PGM=IEBPTPCH
//SYSPRINT    DD SYSOUT=A
//SYSIN       DD   *
  PRINT PREFORM=FBA
//SYSUT2      DD SYSOUT=*
//     *SYSUT1 CAN BE PS OR MEMBER OF A PDS THAT RESIDES ON THE
//     *MVS SYSTEM
//     *ONE CAN INCLUDE THE PRINT FILE AS INSTREAM DATA AS
//     *SHOWN IN THIS EXAMPLE
//*
//SYSUT1      DD DATA
  (include print file here)
```

```
//IEBGENER JOBCARD
//*     THIS JOB STREAM RESIDES ON YOUR REMOTE HOST, YOU CAN
//*     SUBMIT THIS JOB TO THE JES INTERNAL READER OF YOUR IBM
//*     SYSTEM BY ENTERING THE FOLLOWING COMMANDS:
//*
//*     FTP MVS  * CONNECT TO THE IBM HOST
//*     USERID/PSW  * LOGIN TO THE IBM HOST
//*     QUOTE SITE SUBMIT  * NOTIFY SNS/TCP SFTP TO
//*     PUT IEBGENER  * SUBMIT JCL TO JES INTERNAL READER
//GENER       EXEC PGM=IEBGENER
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD DUMMY
//SYSUT2      DD SYSOUT=*
//*     SYSUT1 CAN BE PS OR MEMBER OF A PDS THAT RESIDES ON THE
//*     MVS SYSTEM. ONE CAN INCLUDE THE PRINT FILE AS INSTREAM DATA
//*     AS SHOWN IN THIS EXAMPLE
//*
//SYSUT1       DD DATA
  (include print file here)
```

Data Transfer Operations

This section describes the operation of data transfers by the Server FTP.

Transfer Commands

"Table 5-10" on page 5-39 lists options invoked by the **PUT**, **APPE**, and **GET** commands that initiate data transfer:

Table 5-10 Transfer Options

Option	Function
STOR	saves a file on an MVS system
APPE	appends to a file stored on an MVS system
RETR	retrieves a file from an MVS system

A **2**26-Transfer complete reply is sent only when the disk file being written has been closed successfully or when the data being retrieved and sent across your network has been fully acknowledged.

Truncating and Folding Records

The Server FTP generally folds rather than truncates a network record that exceeds LRECL (or BLKSIZE, for an unblocked data set). You can force truncation with an **ALLO R** command.

Raveled Files

A *raveled* file is a file that contains the network data concatenated into the file with no record markers. Such a file can be created and later retrieved with FTP without loss of information. However, it is not usually possible to process it with any IBM utility. The only transformation generally done on a raveled file is to translate between ASCII and EBCDIC for TYPE A.

Create or retrieve a raveled file with the FTP parameters **STRU F** and one of these:

- Character type (TYPE A or TYPE E) and unblocked logical records
- TYPE I or TYPE L

When a raveled file is stored by the Server FTP, it is folded into maximum-sized logical records.

Padding Fixed-Length Records

If the data set has a RECFM containing F (i.e., it has fixed-length logical records), and if it is not raveled, the Server FTP does these steps:

- Pads each record being written to the next LRECL.
- Removes trailing pad characters from each record read from disk.

The pad character is normally blank for character types and zero for binary types. However, the **SITE** command PAD option can set it to a different value. Although this is not strictly invertible, in most cases it saves transmission time and network bandwidth. The statistics at the end of a file transfer contain the number of records padded if this number is nonzero.

Translation

For the ASCII transfer (TYPE AN, TYPE AT, or TYPE AC), EBCDIC data in a disk file is translated to ASCII over the network and vice versa. The Server FTP always stores data on disk in EBCDIC.

Line Image Files

A line image file is defined by FTP parameters **STRU F**, one of the character types (TYPE AN, TYPE EN, TYPE AT, or TYPE ET), and a blocked disk data set (i.e., not raveled). With line image files, end-of-line (EOL) in the network data is mapped to MVS end-of-record and vice versa. Network end-of-line is normally NL (EBCDIC) or CRLF (ASCII). However, CRLF is also recognized in EBCDIC network data. Isolated CR or LF characters are not recognized as an EOL sequence.

Record Structured Files

For the FTP parameter **STRU R**, a network record is mapped into an MVS logical record and vice versa.

If **ALLO R** is specified, each network record that exceeds **ALLO R** is truncated.

With **STRU R**, a storage operation (i.e., **STOR** or **APPE**) that creates a print file (TYPE AT, TYPE AC, TYPE ET, or TYPE EC) requires the data set to be blocked. If the data set is not blocked, the operation fails with the reply

```
501 print type and STRU R requires blocked data set.
```

Tabs

Horizontal tab characters (HT) received from the network are expanded and/or deleted in accordance with the current **SITE** command TABS option setting. Default is a tab every eight columns; this can be overridden by the **SITE TABS** command.

A vertical tab (VT) character is always treated as data and stored in the file.

Carriage Control and Format Effectors

For character types with format

ple locally applied rules that do not require buffering data. These rules are invertible between storing and retrieving data with the same parameters.

Some obscure cases cannot be handled correctly in this way. In fact, to fully define the correspondence, you must compare the effects of the files on assumed printing mechanisms to achieve the same appearance. This requires that FTP define additional attributes (such as the size of a page and the effect of an overstrike).

The rules assume that at the beginning of a file, the ASA line printer is positioned on the first line. Therefore, the default carriage control character used to create the first line of an ASA print file is **+** (suppress space), with following lines normally using a blank (single space) before printing.

Character Type Rules

This section describes the transformation rules for creating and retrieving character-type (TYPE A or TYPE E) files.

In all cases, the network data can be ASCII or EBCDIC, but a disk data set is always EBCDIC. In addition, an HT character in a received file being stored (via **STOR** or **APPE**) is always handled in accordance with the current TABS value.

File Structure with No Format Control

The **STRU F** command with TYPE AN or TYPE EN parameters define a file-structured (line-image) character file with no format control. Generally, with these parameters, network lines are mapped to and from logical records. The network data can be parsed into a series of lines, using the following syntax

text...eol

Syntax Description

text	A (possibly null) block of text.
eol	An end-of-line sequence (CRLF or NL).

Usage Guidelines

- Other format effectors, including isolated CR and LF characters, can be included in text.

Line Image Files

If the data set to be written or retrieved is blocked (RECFM includes B), the file is assumed to contain line images.

Storing Line Image Files

These rules apply to line image files being stored (via **STOR** or **APPE**):

- An ASCII file (TYPE A) is translated to EBCDIC.
- The text is scanned for the EOL sequence (CRLF or NL).
- Each line is mapped into an MVS logical record, and the EOL sequence is discarded.
- A line that exceeds the target file LRECL is folded into subsequent logical record(s) rather than being truncated.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- If the data set has fixed length records (RECFM includes F), each line is padded with blanks (or the PAD character specified via the **SITE** command) to the next logical record boundary.
- Any control characters (including format effectors but not including the EOL sequence) are left in the data stream.
- If the data set has ASA carriage control (RECFM includes A), a blank Carriage Control Character (CCC)

is inserted at the beginning of each logical record.

Retrieving Line Image Files

These rules apply to line image files being retrieved (via RETR):

- Each MVS logical record is mapped into a line with an EOL sequence inserted after each line.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has fixed-length logical records (RECFM includes F), all trailing blanks (or the PAD character specified via the **SITE** command) are stripped off.
- If the data set has ASA carriage control (RECFM includes A), an additional transformation is applied. If the ASA CCC is the first character of the logical record and *text* is the rest, the data shown in "Table 5-11" on page 5-43 is sent over the network:

Table 5-11 Retrieving Line Image Files Using Carriage Control Characters

CCC	Network data
blank	eol *text*
0	eol eo**l** *text*
-	eol eol eol *text*
1	CR FF *text*
+ (first record)	*text*
+ (other)	CR *text*

A single logical record can result in a series of network lines, and a **+** CCC in the first record of the file is effectively ignored.

Raveled Files

If the data set to be written or retrieved is unblocked (RECFM does not include **B**), the file is assumed to be a raveled file and is treated as a single-byte string.

Storing Raveled Files

The rules in this list apply to a raveled file being stored (via **STOR** or **APPE**).

The data set stored on disk is a concatenation of the network data. That is, the network data is folded into logical records.

An ASCII file (TYPE A) is translated to EBCDIC.

Retrieving Raveled Files

These rules apply to a raveled file being retrieved (via RETR):

- The network data set is a concatenation of the retrieved logical records. That is, the logical records retrieved from disk are raveled into a single stream of bytes for transmission across the network.
- If the data set has ASA carriage control (RECFM includes A), the CCC is deleted and is not included in the network data.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.

File Structure with Telnet Format

The **STRU F** command with TYPE AT or TYPE ET parameters define a file-structured character file with Telnet format effectors. Generally, with these parameters, a network line maps to/from a logical record. The network data can be parsed into a series of segments using the following format:

```
fe text
```

Syntax Description

fe	A sequence of ASCII format effectors.
text	A (possibly null) block of text.

Usage Guidelines

- Data storage with these parameters follows the same rules as storage of **STRU F**, TYPE AN or TYPE EN, with one exception. In the line image case, if the data set being stored into (via **STOR** or **APPE**) has ASA carriage control (RECFM includes A), the CCC is not set to blank (as in the **STRU F**, TYPE AN or TYPE EN case) but is determined from the network data according to the table in "Retrieving Line Image Files" on page 5-43. Any format effectors not involved in this transformation are left in the data stream.
- Retrieval of data using these parameters follows the same rules as retrieval of **STRU F**, TYPE AN or TYPE EN.

File Structure with ASA Format

The **STRU F** command with TYPE AC or TYPE EC parameters define a file-structured (line-image) print file containing ASA carriage control. Generally, with these parameters, a network line maps to/from a logical record. The network data consists of these parameters:

cc text...eol

Syntax Description

cc	An ASA Carriage Control Character.
ext	A (possibly null) block of text.
eol	An end-of-line sequence (CR, LF, or NL).

Line Image Files

With these parameters, if the data set to be written or retrieved is blocked (RECFM includes B), the file is considered to be line images.

Storing Line Image Files

These rules apply to line image files being stored (via **STOR** or **APPE**).

- An ASCII file (TYPE A) is translated to EBCDIC.
- The text is scanned for the EOL sequence (CRLF or NL).
- Each line is mapped into a logical record, and the EOL sequence is discarded.
- A line that exceeds the target file LRECL is folded into subsequent logical record(s), and a warning message is issued. If the file being written has ASA carriage control, each of the subsequent logical records is stored with a blank CCC.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- If the data set has fixed length records (RECFM includes F), each line is padded with blanks (or the PAD character specified via the **SITE** command) to the next logical record boundary.
- Any control characters (including format effectors but not including the EOL sequence) are left in the data stream.
- If the data set has ASA carriage control (RECFM includes A), the ASA CCC from the network data is used as the CCC for the stored logical record. If the data set does not have ASA carriage control, the first character of the network record is not stored in the logical record.

The Server FTP writes such a print file only into a blocked data set (RECFM includes B).

Retrieving Line Image Files

When a **RETR** operation is performed with these TYPE and STRU parameters from a blocked data set, a new line image file is created and sent over the network. These rules apply to line image files being retrieved (via **RETR**):

- Each MVS logical record is mapped into a line with an EOL sequence inserted after each line.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has fixed length records (RECFM includes F), all trailing blanks (or the PAD character specified on the **SITE** command) are stripped off before sending on the network.
- If the retrieved data set contains ASA carriage control (RECFM includes A), the CCC from the logical record is passed as part of the network record. Otherwise, a blank CCC is inserted at the beginning of each network record, except for the first network record into which a **+** CCC is inserted.

Raveled Files

If the data set to be written or retrieved is unblocked (RECFM does not include **B**), the file is considered to be a raveled file and is treated as a single string of bytes.

Storing Raveled Files

The Server FTP does not write a raveled (unblocked) file with these data transfer parameters.

Retrieving Raveled Files

These rules apply to a raveled file being retrieved (via **RETR**):

- The network data set is a concatenation of the retrieved logical records. That is, the logical records retrieved from disk are raveled into a single stream of bytes for transmission across the network.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has ASA carriage control (RECFM includes A), the CCC of the first logical record is included in the network data. Any other carriage control is deleted. If the data set does not have carriage control, a **+** CCC is inserted at the beginning of the network data, and no other carriage control is inserted.

Record Structure with No Format

The **STRU R** command with TYPE AN, TYPE EN, TYPE AT, or TYPE ET parameters define a character file with record structure. Generally, network records are mapped to/from logical records.

fe text...eor

Syntax Description

fe	A (possibly null) sequence of ASCII format effectors.
text	A (possibly null) block of text.
eor	An end-of-record sequence.

Storing Logical Records

These rules apply to files being stored (using **STOR** or **APPE**) with these parameters:

- An ASCII file (TYPE A) is translated to EBCDIC.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- A network record longer than **ALLO R** (if specified) is truncated, and a warning message is issued.
- If a network record (after possible **ALLO R** truncation) is longer than LRECL, it is folded into subsequent logical record(s) and a warning message is issued.
- If the data set has fixed length logical records (RECFM includes F), each network record is padded with blanks (or the PAD character specified via the **SITE** command) to the next logical record boundary.
- If the data set has ASA carriage control (RECFM includes A), the CCC is normally blank, and any embedded ASCII format effectors are treated as data. However, if the TYPE is AT or ET, the escape sequences shown in "Retrieving Logical Records" on page 5-47 (F and R) are scanned for and used to set CCC if found in the network data. No scanning for the escape sequences occurs with TYPE AN or TYPE EN.

The Server FTP writes a print file with Telnet format effectors (TYPE AT or TYPE ET) only into a blocked data set (RECFM includes B).

Retrieving Logical Records

These rules apply to files being retrieved (via **RETR**) with these parameters:

- Each logical record (RECFM includes B) or each physical record (RECFM does not include **B**) is mapped into a network record. If the data set is unblocked (raveled), the network data is a concatenation of the logical records. The other transformations described in the following steps still apply.
- If the data set has fixed length logical records (RECFM includes F), all trailing blanks (or the PAD character specified via the **SITE** command) are stripped off before data is sent on the network. This occurs even for unblocked data sets. A completely blank (or null) record is sent as a null record with the appropriate end of record sequences based on mode.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has ASA carriage control (RECFM includes A), an additional transformation is applied. "Table 5-12" on page 5-47 shows the transformations that map the CCC in the disk data set into network data. A one-to-many relationship can occur between disk records and network records:

Table 5-12 Retrieving Logical Records Using Carriage Control Characters

CCC	Network Data
blank	*text eor*
0	*eor text eor*
-	*eor eor text eor*
1	F' *text eor*
+ (first)	R' *text eor*
+ (other)	*text eor*

Note: If the type is ASCII (TYPE AN or TYPE AT), the EBCDIC single quote (**'**) escape character is translated to an ASCII backslash ****.

Record Structure with ASA Format

The **STRU R** command with TYPE AC or TYPE EC parameters define a record-structured print file containing ASA carriage control. Generally, with these parameters, network records map to/from logical records.

cc text...eor

Syntax Description

cc	An ASA CCC.
text	A (possibly null) block of text.
eor	An end-of-record sequence.

Storing Print Files

These rules apply to files being stored (via **STOR** or **APPE**) with these parameters:

- An ASCII file (TYPE A) is translated to EBCDIC.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- A network record longer than **ALLO R** (if specified) is truncated, and a warning message is issued.
- If a network record (after possible **ALLO R** truncation) is longer than LRECL, it is folded into subsequent logical record(s) and a warning message is issued. If the file being written has ASA carriage control (RECFM includes A), each of the subsequent logical records is stored with a blank CCC.
- If the data set has fixed length logical records (RECFM includes F), each network record is padded with blanks (or the PAD character specified via the **SITE** command) to the next logical record boundary.
- If the data set has ASA carriage control (RECFM includes A), the ASA CCC from the network data (cc) is used as the CCC for the stored logical record. Otherwise, the first character of the network data is deleted and is not stored in the logical record.

The Server FTP writes such a print file only into a blocked data set (RECFM contains B).

Retrieving Print Files

These rules apply to files being retrieved (via RETR) with these parameters:

- Each logical record (RECFM includes B) or each physical record (RECFM does not include B) is mapped into a network record. If the data set is unblocked (raveled), the network data set is a concatenation of the logical records. The other transformations described in the following steps still apply.
- If the data set has fixed length logical records (RECFM includes F), all trailing blanks (or the PAD character specified via the **SITE** command) are stripped off before sending on the network. This occurs even for unblocked data sets. A completely blank (null) record is sent as a null record with the appropriate end of record sequences based on mode.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the retrieved data set has ASA carriage control (RECFM includes A), the CCC from the logical record is passed as part of the network record. Otherwise, a blank CCC is inserted at the beginning of each network record, except for the first record, into which a **+** CCC is inserted.

Binary-Type Rules

This section describes the transformation rules for creating and retrieving binary-type (TYPE I or TYPE L) files. The term “image-type” is used interchangeably with “binary-type” in the information in this section.

No Record Structure

The **STRU F** command with TYPE I or TYPE L parameters define a binary file without record structure.

Storing Binary Files

These rules define the transformations to such a file when it is stored (via **STOR** or **APPE**) by the Server FTP:

- The received network data is folded into maximum-length logical records when stored by the Server FTP.
- If a null (zero-length) network record is received, it is ignored.
- No translation, truncation, or padding is performed on the data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not write image-type data into a print data set (RECFM includes A).

Retrieving Binary Files

These rules define the transformations to such a file when it is retrieved (via **RETR**) by the Server FTP:

- The retrieved logical records are concatenated into a single stream of network data.
- No translation, truncation, or removal of padding is performed on the retrieved data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not retrieve from a print data set (RECFM includes A).

Record Structure

The **STRU R** command with TYPE I or TYPE L parameters define a binary file with record structure.

Storing Structured Binary Files

These rules define the transformations to such a file when it is stored (via **STOR** or **APPE**) by the Server FTP:

- If a network record with zero data bytes is received, it is stored as all pad characters if RECFM includes F, zero data bytes if RECFM includes V, and x'00' if RECFM is U.
- A network record longer than **ALLO R** (if specified) is truncated to **ALLO R**, and a warning message is issued.
- If a record (possibly truncated to **ALLO R**) is longer than the maximum logical record length, it is folded into multiple logical records, and a warning message is issued.
- If the data set has fixed-length logical records (RECFM includes F), each network record is padded with zeros (or the PAD character specified via the **SITE** command) to the next logical record boundary.
- No translation is performed on the data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not write image-type data into a print data set (RECFM includes A).

Retrieving Structured Binary Files

These rules define the transformations performed on such a file when retrieved (via **RETR**) by the Server FTP:

- The retrieved logical records are mapped into network records.
- If the data set has fixed length records (RECFM includes F), all trailing zero bytes (or the PAD character specified via the **SITE** command) are stripped off before sending on the network.
- No translation is performed on the retrieved data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not retrieve from a print data set (RECFM includes A) for transmission as image-type data

Non-Invertible Retrieval

Unless a file is raveled, storing it with the Server FTP might transform the data in ways that are not strictly invertible. (That is, if the file is later retrieved using the same parameters, it might not be identical to the original file sent to this Server FTP.) The file would generally mean the same thing, but some of its byte stream is changed.

Sources of Non-Invertibility

Here are some of the sources of non-invertibility introduced when a file is stored by this Server FTP:

- All files

If the data is stored into a data set with fixed-length blocks and if a record/line ends with the pad character the Server FTP uses to pad the block, that pad character is stripped off when the file is retrieved.

- Character files

HTs might be expanded to blanks during storage and not be reintroduced when the file is retrieved.

LF and FF, which do not appear at the left margin, insert blanks to position the virtual print head correctly. On RETR, these blanks and the extra CRLF that were not in the original input are returned to the remote host.

NL is used interchangeably with CRLF in an EBCDIC file. Therefore, CRLF received in such a file is sent out as an NL.

LFs preceding an FF with no intervening data are removed.

Other Features

This section lists some other useful features of the FTP program.

Testing and Debugging

The STAT display includes the Internals section that contains a number of important control fields and pointers. This information is generally of interest only to the system programmer.

The TCP-level tracing and debugging facilities are described in the *TCPaccess Customization Guide*.

Telnet Break

The Telnet commands **IP**, **BRK**, or **CONTROL-C** on the control connection stops output from commands such as **HELP** and **STATUS**.

6

Client/Server Telnet

This chapter describes the TCPaccess Telnet facilities. It provides the information necessary to develop a working knowledge of the TCPaccess implementation of Client Telnet and Server Telnet. This chapter contains these sections:

- "Introducing Client/Server Telnet" on page 6-2

  Provides a brief overview of the services Client Telnet provides.

- "Client Telnet" on page 6-3

  Describes how to use the Client Telnet facilities to access TCPaccess locally.

- "Server Telnet" on page 6-18

  Describes how to use the Server Telnet facilities to access TCPaccess remotely.

- "Telnet Escape Sequences" on page 6-20

  Describes the escape sequences TCPaccess uses to implement the Telnet protocol.

Introducing Client/Server Telnet

TCPaccess provides Client Telnet facilities that let TSO and VTAM users access your network through the Telnet protocol. TCPaccess also provides Server Telnet facilities that let remote network users access host application programs. These facilities are shown in “Table 6-1” on page 6-2.

Table 6-1 Telnet Facilities

User-level Protocol	Service	Notes
Server Telnet	Lets network users access TSO and other VTAM applications.	Application must support 3278 or 3767 through VTAM.
	Provides TCPaccess help, news, status display, etc.	Implemented within TCPaccess.
Client Telnet	Lets MVS users access network hosts.	Supports access from 3278s and 3767s via VTAM and from TSO via the **TELNET** command.

Client Telnet

The Client Telnet facilities provide local access to your network via TCPaccess. You can access TCPaccess locally in either of these ways:

- Indirectly, using the Client Telnet (or FTP) command processors under TSO

  The TSO Telnet processor supports line-by-line and full-screen terminals, but both are mapped into line-by-line Network Virtual Terminal (NVT) operation of the remote host. A difficulty with this program is that TSO supports only half-duplex locked-keyboard operation, which makes access to character-by-character hosts awkward. You can receive pending screen data only by using Enter.

  The TSO Telnet program is written in PL/I and you must have the PL/I Transient (runtime) Library (an IBM program product). Some useful features of this program are saving typescripts and multiplexing several sessions at once.

- Directly, using VTAM-supported 3767 or 3278 terminals

  Although the TCPaccess 3278 terminal manager does not have all the features of the TSO Telnet program, it can access the 3278 with a true full-duplex unlocked-keyboard protocol.

  The 3278 can either be mapped into line-by-line operation as an NVT or can be operated in transparent 3278 full-screen mode to access a remote IBM MVS or VM server.

The TSO Client Telnet Command

The TSO Client Telnet command, **TELNET**, provides a TSO Client Telnet interface to the network. The **TELNET** command supports only line-by-line or NVT operation of remote hosts. However, it has functions, such as multiple concurrent sessions, that the direct VTAM Client Telnet lacks.

Client Telnet operates the local terminal in either line or screen mode, depending on whether it is accessed from a terminal of the IBM 3270 family or from an ASCII terminal. Choice of mode is automatic and usually transparent. However, you can override the automatic choice if you need to operate in line mode on a terminal. This may prove useful if you use facilities such as the Session Manager in TSO/E.

Since IBM systems normally do not support character-by-character interactions, Client Telnet does not operate in character-oriented mode, and it can be inconvenient to communicate with processes on remote hosts that do operate in such a mode. Because there may be such hosts on a network, Client Telnet implements devices and techniques to ease the incompatibility.

The TSO Telnet program screen mode can present one or more multiplexed line-oriented terminal sessions; however, full-screen interaction with a processing program is not possible with this version of the program.

In screen mode, Client Telnet does all its own screen management. Client Telnet is not compatible with operation under the IBM ISPF program product. It can be invoked under ISPF, as can most TSO command processor programs, but Client Telnet is not aware of the ISPF environment, so it does not support such ISPF features as split-screen operation. In anticipation of future enhancements, this version of Client Telnet reserves certain screen fields and function keys for ISPF compatibility.

Some options of the TSO Telnet program (the **PRINT** and **TEST** commands) require allocation of a SYSPRINT file, but this is not absolutely necessary in normal operation (that is, when you are not using **PRINT** or **TEST**).

Allocating a SYSPRINT file to the terminal in screen mode causes constant switching between screen and line modes. To avoid this, allocate the SYSPRINT file to a SYSOUT file instead of to the terminal.

The TELNET Command

Invoke the TSO Telnet program with the TSO **TELNET** command in one of these forms:

TELNET

TELNET / *argument argument*...

No arguments are required and none are useful to most Client Telnet users. Any that are specified must be preceded by slash (/) to accommodate the conventions of the PL/I runtime support package.

The two classes of **TELNET** command options, general and debugging, are described in the following sections.

General Command Options

These are the general Telnet command options:

- TTY

  The TTY option specifies that your terminal is capable of generating carriage returns. Since virtual 3767 line terminals such as those supported by the Virtual Line Terminal (VLT) facility do not generate carriage return (CR) or new line (NL) characters at the end of lines, Client Telnet automatically appends an NL to every line of user input that is received. The TTY option disables this by specifying to Client Telnet that your terminal appends either a CR or an NL at the end of every line of input. This option is useful in supporting real local ASCII terminals that connect to TSO through TCAM, NTO, or NPSI.

- LINE

  The LINE option causes the TSO Telnet program to drive the terminal in line mode even if the terminal is a CRT.

- SYS

  The SYS option, in the form SYS=*x*, where *x* is an arbitrary character, causes Client Telnet to open its VLT connection to a network name of ACCESx instead of the usual ACCES. This allows communication through a test version of TCPaccess.

- APPLID

  The APPLID option, in the form APPLID=*aaaaaaaa*, where *aaaaaaaa* specifies to the TSO Telnet program the default VTAM application ID of the local TCPaccess. This command causes Client Telnet to open its VLT connection to a network name of *aaaaaaaa* instead of the usual ACCES. If supplied, this parameter need not point exclusively to the local TCPaccess; it can refer to any VTAM application. For example, TSO is the necessary APPLID to connect to TSO.

Debug Options

The debugging options, TEST and U, are described here:

- TEST

  The TEST option causes the program to operate in test mode, where status information is written to the SYSPRINT file. This information is essentially unformatted and is not useful to the casual Telnet user.

- U

  The U option, specified as U=*userid*, modifies the output of TEST mode. It arranges to send the output via **TPUT** to the specified TSO user ID instead of to SYSPRINT.

TSO Client Telnet Operation

Once the program has been activated, you can enter Telnet commands or data to be transmitted on the session. In session data, the logical not character (¬) is reserved as an escape character. To transmit the ¬ character, you must type it twice (¬¬). Refer to “Telnet Escape Sequences” on page 6-20 for details on using the Telnet escape character.

Line Mode Operation

TSO Telnet operates in line mode if TSO believes your terminal is line-oriented or if you have used the LINE argument when invoking the program. In most cases, Client Telnet commands operate essentially the same in either line or screen mode.

The techniques used to send data lines in line mode are described here:

- The keyboard operates in unlocked mode; you can type at any time. However, the underlying TSO system is operating in half-duplex, line-by-line mode, so sometimes your typing might be interrupted. If Client Telnet interrupts your typing with output, you must retype the portion that was not successfully read. It is not always possible to tell which characters were read and which were not, so it is best to abort the input line and start over. Since you usually know when output is expected and when it is not, this should not be a problem.
- In line mode, the data you type is sent to the currently active session unless the first character is a greater than (**>**) symbol (in which case it is sent to the TSO Telnet program rather than to the currently active session) or there is no session active. In either case, the data is taken to be a Telnet command.
- When you terminate an input line (including a null line) with a carriage return, the data is sent with a new line character appended.
- When you terminate an input line with CONTROL-D, the data is sent without a new line character. This facilitates communication with remote systems that operate in character mode.
- CONTROL-C and ESCAPE can be used as data characters and are transmitted properly. Most other control characters are filtered from the input by TSO. CONTROL-C is usually interpreted by a remote IBM process as an attention.
- The ATTN key is reserved for use by the local process. It stops output flow long enough to enter an input line.

Screen Mode Operation

TSO Telnet operates in screen mode if your terminal is believed by TSO to be a display terminal of the 3270 family and if you have not used the LINE argument when invoking the program. The rules of screen mode interaction seem complex, but screen mode is very useful.

In screen mode, the screen is divided into the following areas:

- The TSO Telnet banner and version number

  This field is also used to present short error or exception messages.

- The primary input area

  This 149-character field (CMD) is where you type most session input and Telnet commands, so TSO Telnet keeps moving the cursor to the beginning of this field.

- The command input area

  This field is provided for future ISPF compatibility and is not needed for normal TSO Telnet operation.

- The current VTAM application identification default

  This field points to the local TCPaccess application. Every Telnet **CONNECT** command initiates a VTAM connection to the TCPaccess currently identified by the APPLID default. Multiple VTAM sessions to different VTAM applications are possible by changing the APPLID default dynamically. This is described in “VTAM Client Telnet” on page 6-14.

- The current session identification

  When a session has been established, this identifies the session number and the host to which it is connected.

- A list of other session numbers

  Each session number is a single symbol, hence the limit of ten concurrent sessions. If a session is defined but not currently selected, its symbol **I** appears in this list. If the session has output waiting, its symbol **O** appears in the list. Undefined sessions are shown as a period (**.**).

- A separator row of dash characters (**-**)

  This row conceals a set of indicators that are replaced as various operating modes are activated. These are the operating modes:

  – AUTO for automatic page turning
  – NOECHO for non-display of output
  – READ during read processing
  – WRITE during write processing
  – SLEEP when the keyboard is disabled
  – TEST during test debugging
  – HIDE when the input line is turned into a non-display line

- 17 rows of output area

  Both input and output data are echoed to the output area. This data is not scrolled; the current output line is indicated by a row of equal characters (=). The line of equal characters rolls around the screen, erasing old output and overlaying it with new. When the indicator wraps from the bottom of the output area to the top, press Enter to prevent overwriting data that has not been read (i.e., “turn the page”).

- 2 rows of Program Function key (PF) definitions, associating certain commands with function keys

  These associations are fixed and cannot be overridden. The key assignments are chosen to be compatible with the default assignments used by the ISPF program product, and, in this version, some function keys are reserved for ISPF functions that have not yet been implemented (SPLIT and SWAP). Like ISPF, Client Telnet follows the convention that there are no functions available through function keys that are not also available through commands.

  In screen mode, you normally type into the input area; however, you can modify lines in the output area and cause them to be reread as input. Read "Retransmitting Data" on page 6-8.

The NULL Transaction

In most cases, Client Telnet commands operate the same in either line or screen mode. However, the techniques used to send data lines and to differentiate them from command lines are different. Because IBM terminals operating under TSO are half-duplex, it is not possible to operate in screen mode with an unlocked keyboard.

For a NULL transaction in screen mode, press Enter with no screen fields modified or with all modified fields blank for a no-operation. This does not send data; it merely returns control of the terminal to the TSO Telnet program and allows the program to switch into output mode.

The most obvious effect of this is that an empty line can be sent ***only*** by using a Telnet command (XWNL) or a function key (PF10, which sends a NULL transaction and a new line (NL)).

A more important effect is that communication in screen mode frequently requires constantly using Enter to keep output flowing. Client Telnet tends to hold control of the terminal until there is an indication that no more output is immediately available. You can control how long the program waits for this indication, but the defaults are satisfactory for most conditions. While the null transaction is used frequently in screen mode, the real work of a Telnet session is done with the other kinds of transactions. The most common occur when non-blank data is typed into the input field and/or when the keyboard is locked through a key other than Enter.

Sending Data

There are several ways to send data to the current session. The usual method is to type a complete input line into the input area and press Enter, which stands for the **SEND** command. The data is sent, including explicit leading and trailing blanks, and a new line character is sent after them.

Another method is to type the **SEND** command (or one of the other similar commands) into the input area, followed by the data to be sent, and then press PF5 instead of Enter. PF5 stands for the **EXEC** command, which causes Client Telnet to parse its input into a command and an operand string, and then to execute the command. The operand string begins with the character after the single blank that terminates the command.

A third method is to type the data into the input field and press a function key that has been assigned the value of a command. This is equivalent to using an explicit command and PF5 **(EXEC)**.

With any of these methods, you can enter a command name in the command field on the screen. That command takes precedence over the command implied by the key you press. Normally, you do not need to use this method of input, but the rules of interaction with ISPF make it necessary.

Command Entry Rules

Many Client Telnet commands perform functions other than sending data. In all cases, the same rules apply.

- If you clear the screen and enter the command at the top left of the screen, the key you use has no effect. The command is parsed for a command verb and operands. (This is compatible with a panic exit, such as **CLEAR** followed by **END**.)
- If you modified the CMD field, its contents are the command verb. Otherwise, the verb is implied by the key you pressed.
- Operands for the command are in the CMD field.
- Any command of the class that transmits data is not active if there is no current session. Such a command is treated as an implied **EXEC**. This means that if there is a current session, strings entered with Enter are data to be transmitted. If there is no current session, strings entered with Enter are commands.
- The **>** convention sends Telnet commands as if PF**5** were pressed. When used in line mode it does not apply to keyboard input in screen mode. (Read the description of the **READ** command in "Commands for Controlling Input and Output" on page 6-10 for a situation in which it does apply.)

Retransmitting Data

You can edit and retransmit lines from the output area of the screen. However, each line of the output area is a screen field and the folding of received data to accommodate the 79-character usable screen width might split a logical line into two separate fields. To make the feature easier to use, PF4 has been assigned the **CURSOR** command. The **CURSOR** command moves the physical cursor to the beginning of the last line echoed into the output area.

These are the rules of transmission from the output area:

- For every user interaction, there is an implicit command that is determined either by the contents of the CMD field or by the function key pressed. (The PA and Enter keys are included for this purpose.)
- For every such interaction, there is an ordered set of modified data fields, potentially including the input area and each line of the output area. The ordering is from the top of the screen down.
- The implied command is applied once with the operand field composed of the concatenation of all the modified data fields. New line characters are not automatically inserted when two lines are concatenated. Use the **-** (dash) command to insert them.

Function Keys

A fixed mapping of commands onto function keys is used in this version of Client Telnet. The commands that can be executed by pressing function keys are listed in "Table 6-2" on page 6-8:

Table 6-2 Client Telnet Function Keys

Function Key	Associated Command	Function Key	Associated Command
PF1	**HELP**	**PF7**	**XESC**
PF2	No action	**PF8**	**XCTL**
PF3 *	**END/BYE**	**PF9**	**HIDE**
PF4	**CURSOR**	**PF10**	**XWNL**

Function Key	Associated Command	Function Key	Associated Command
PF5	EXEC	PF11	XNNL
PF6 *	BRK/KO	PF12	RETURN
Enter	SEND	PA1 *	ATTN/IP
		PA2	RESHOW

Note: Those commands annotated with an asterisk can be entered when the screen shows the message **HIT ENTER TO TURN PAGE**.

TSO Client Telnet Commands

This section outlines commands for sending data, for session control, for controlling input and output, and miscellaneous commands.

Some commands can be entered by hitting a programmed function key. Where a function key can be used in place of typing the command, the key name is shown to the right of the command name.

Commands for Sending Data

These are the commands for sending data:

- **SEND** or **ENTER**

  The **SEND** command sends its operand followed by a new line. If the operand is null, **SEND** has no effect (no-op).

- **XWNL** or **PF10**

  The **XWNL** transmit with null line command sends its operand followed by a new line. A null operand results in only a new line.

- **XNNL** or **PF11**

  The **XNNL** transmit with no null line command sends only its operand. If the operand is null, **XNNL** is a no-op.

- **XCTL** or **PF8**

  The **XCTL** transmit control command sends its operand after transforming only the last character into control case (CONTROL-x). If the operand is null, **XCTL** is a no-op.

- **XESC** or **PF7**

  The **XESC** transmit escape command sends its operand followed by an escape. If only an escape is sent, a null operand results.

- **KO** or **PF6**

  The **KO** kill output command transmits an abort-output signal followed by an interrupt-process signal. Any operand is ignored.

- **HEX**

  The **HEX** command interprets its operand as hexadecimal field characters, so the operand must consist of an even number of hexadecimal digits with no blanks or delimiters. The operand is converted to binary bytes and transmitted. The TSO Telnet program operates in EBCDIC mode and translates your string from EBCDIC to ASCII before placing it on the network.

- **IP** or **PA1**

  The **IP** command transmits an interrupt-process signal. Any operand is ignored.

  The PA1 function key is also used for **ATTN** command.

- **BRK** or **PF6**

  The **BRK** command transmits a break signal. Any operand is ignored.

Commands for Session Control

These commands are sent to Client Telnet to add sessions, switch between sessions, and change the status of a session:

Note: Any matching function keys are in parentheses to the right of the command.

- **APPLID** *string*

  Changes the current APPLID default used by TSO Telnet to connect to the local TCPaccess. The new APPLID is given by *string*. Thus, connections to multiple copies of TCPaccess or to other VTAM applications are possible.

- **END | BYE (PF3)**

  Terminates the current TSO Telnet activity. Normally, this is a session or a HELP screen. However, if no sessions are defined and **HELP** is not in effect, TSO Telnet is terminated. This command is refused if you issue it when there are sessions defined but no session is current.

- **RETURN (PF12)**

  Ends all TSO Telnet activity. It is equivalent to multiple **END** commands.

Commands for Controlling Input and Output

These commands manipulate the TSO Telnet session that is currently running:

- **TTO** *number*

  This command is effective only in screen mode. It specifies the number of milliseconds that TSO Telnet is to wait for more output before unlocking a locked keyboard. Large values cause sluggish operation, and small values require excessive use of **ENTER**. The default value of 500 is a reasonable compromise in most cases.

- **READ** *dsname* **| OFF**

  The **READ** command opens the file *dsname* and reads its records as Telnet input lines. Each line is processed as though entered from the keyboard with the **ENTER** or carriage return in line mode. This

means that if you are operating in screen mode, lines beginning with the greater than (>) character are processed as though entered with PF5, that is, as Telnet commands.

Use this feature carefully, because it can cause confusing results if errors occur.

The *dsname* is specified in the usual TSO syntax, either quoted or unquoted. It can name any file that can be read sequentially with the PL/I READ statement, including a PDS member. Blanks and sequence numbers are treated as data by the TSO Telnet **READ** command, so an unsequenced variable-length file is the preferred input form. TSO Telnet continues to accept input from the screen during the **READ** operation. Lines read from the screen are executed at whatever point in the file they fall. The most useful application of this feature is the ability to enter the **READ OFF** command to abort reading. Otherwise, reading proceeds to end-of-file and stops. Obviously, you cannot read a file named OFF unless you use its quoted name. When a **READ** operation is in effect in screen mode, a **READ** indicator is visible on the separator line.

- **RTO** *seconds*

  The Read Timeout command specifies the number of milliseconds that TSO Telnet is to wait between input records during a **READ** operation. Normally, a **READ** operation is limited by the rate at which the data can be transmitted or by the need to turn the page as the read data is echoed; however, this command is provided for cases where those limitations do not apply. The default value is 500 milliseconds.

- **WAIT** *milliseconds*

  This command causes TSO Telnet to pause just once for the number of milliseconds specified. You can interleave your READ data with **WAIT** commands at points where you know an operation takes a lot of time. For instance, it is wise to include **WAIT** commands behind **CONNECT** commands and LOGON sequences.

- **WRITE dsname OFF**

  Use the **WRITE OFF** command to terminate writing. You cannot write to a file named OFF unless you use its quoted name. When a **WRITE** operation is in effect in screen mode, a **WRITE** indicator is visible on the separator line.

  WRITE OFF opens the file *dsname* and echoes Telnet input and output records into it. This produces a typescript of all interactions with all sessions that TSO Telnet is managing. The named file must already exist. Its DCB characteristics are changed to VB, 260, 4000.

- **PRINT**

  This command is effective only in screen mode. It writes a snapshot of only the current screen to the SYSPRINT file. It is different from **WRITE**, which writes continuously into a data set of your choice.

- **ECHO ON | OFF**

  This command controls echoing of output to the terminal and, in screen mode, echoing of input to the output area. When **ECHO OFF** has been specified, no output is written to the terminal, and a **NOECHO** indicator is visible on the separator line. This mode is usually used in conjunction with **WRITE**.

- **SAMPLE**

  This command is used in **ECHO OFF** mode. It causes a small amount of output data to be echoed. In screen mode, one page is echoed, while in line mode the sample is determined by the size of the data records being received from the network and is usually only a partial line. **SAMPLE** lets you monitor a session that is writing its output only to a file. However, your **SAMPLE** commands appear in the output file.

- **AUTO ON | OFF**

  This command controls automatic page turning in screen mode. When **AUTO ON** (or just **AUTO**) has been specified, pages are turned without your intervention and an **AUTO** indicator is visible on the separator line. This mode, used with **SLEEP** mode, removes the terminal entirely from your control. Avoid using this combination.

- **SLEEP**

  This command disables keyboard input (so you do not need to keep pressing ENTER to maintain output flow) and places a **SLEEP** indicator on the separator line. The only way to exit **SLEEP** mode is to press **ATTN** (**PA1** in screen mode). This mode can be used with **AUTO** mode to remove the terminal entirely from the user's control. Use caution with this option.

- **ATTN (PA1)**

  This command is always invoked through **ATTN**. In screen mode, attention is signaled through PA1 and is used only to break **SLEEP** mode. In line mode, attention interrupts TSO Telnet operation and requests a new input line. This input is then processed like any other.

  The PA1 function key is also used for the **IP** command.

  Note: The attention key on a 3278 is the PA1 key, not the ATTN key.

- **NOTE**

  This command introduces a limited comment. No data is transmitted, but when you are in screen mode, data is echoed to the output area.

- **HIDE (PF9)**

  This command causes the next input on the terminal not to display. It implements password protection. When in screen mode, **HIDE** causes **HIDE** to appear on the separator line and turns the primary input area into a nondisplay field. It is a toggle switch, so if it is on, you can enter it again to turn it off. Any operand associated with **HIDE** is ignored.

Miscellaneous Commands

These are some miscellaneous commands that can be useful:

- **EXEC (PF5**)

  This command is always invoked through PF5, although it works as a command. It executes its argument as a Telnet command.

- **HELP (PF1)**

  This command presents brief tutorial information. In line mode, it lists common commands briefly. In screen mode, it displays a sequence of HELP screens. You can step through the screens with ENTER or return to Client Telnet with PF3.

- **RESHOW (PA2**)

  This command is meaningful in screen mode only and is invoked through PA2. It restores the screen to its previous condition.

- **CURSOR (PF4)**

  This command is effective in screen mode only. It moves the cursor to the beginning of the last line written to the output area.

- **CLEAR**

  This command is effective in screen mode only. It clears the screen for the current session and resets the current output line to the top of the output area, which can be useful in keeping things together on one display screen.

- **TEST ON | OFF**

  This command controls the output of debugging information. When **TEST ON** (or just **TEST**) is specified, diagnostic data are written to the SYSPRINT file and a **TEST** indicator displays on the separator line.

- **LOG** *userid*

  This command modifies the action of **TEST**. *userid* is the target of **TPUT** macro instructions to write the **TEST** output data. The same data is not written to SYSPRINT. This command lets you receive test output in real time, but on another terminal. **LOG** without an operand stops this special behavior.

- **TSO | DO**

  This command executes its argument as a TSO command and pre-empts Client Telnet temporarily. When the command processor returns, if screen mode is in effect, Client Telnet refreshes its screen.

VTAM Client Telnet

TCPaccess is a VTAM primary application and can support 3278 or 3767 terminals with Client Telnet access to your network.

Invoking VTAM Client Telnet

The VTAM Client Telnet command, **VTELNET**, operates in either NVT (line-by-line) mode or in transparent full-screen 3278 mode, depending on the Telnet negotiations initiated by the remote Server Telnet. The choice of mode is automatic and does not normally concern you except for the usage of special function keys (**PF***n*). See "NVT Operation from 3278 Terminals" on page 6-15 for more information on programmable function key assignments.

Invoke **VTELNET** with one of these commands:

ACCES *host_name*

or

LOGON APPLID(ACCES) to DATA (*host_name*)

Note: *host_name* is a required operand.

The command is entered on the system login invitation screen in place of the **LOGON** command used to start a TSO session. Here **ACCES** is the VTAM application name for TCPaccess and the permissible host name is defined in "Using Host Name Strings" on page 1-5 in "" on page 1-1. For example, if you enter **ACCES SRI-NIC** and the message **SRI-NIC PARAMETER UNRECOGNIZED** displays, it means that the **LOGTAB=INTTAB** parameter has not been set correctly during the TCPaccess installation. Report this problem to your TCPaccess site administrator.

Note: The exact VTAM logon command might be different at your installation. If in doubt, contact your TCPaccess site administrator.

VTAM Client Telnet Operation

When you enter the VTAM **LOGON** command, VTAM connects the terminal to TCPaccess. TCPaccess then checks the host name, displays appropriate error messages, and disconnects the terminal. If the host name is correct, TCPaccess prompts you for a user ID and password, checking that you are allowed access to the network. If you supply the correct user ID and password, the Telnet connection to the remote server is established and the server's banner message displays.

NVT Operation from 3278 Terminals

While the remote host is operating in normal Telnet ASCII or NVT mode, VTAM Telnet maps line-by-line data onto the local 3278 screen much like TCAS does for TSO.

Each line segment is placed on a blank screen sequentially at the cursor. The control characters BS, CR, and LF affect the cursor in the applicable fashion. Characters typed on the keyboard display and transmit with the carriage return/line feed symbol (CRLF) appended when you press ENTER. The necessary 3278 orders are added to the screen data and deleted from the keyboard data.

The terminal operates in full-duplex local echo mode when controlled by VTAM Telnet. You need not press ENTER or any other key to poll for output. Any data received is immediately displayed. If you attempt to type while data is being sent to the screen, the typed data is lost. This is not normally a problem because usually you must wait for a prompt before beginning to type.

The screen can be erased at any time by pressing CLEAR. This homes the cursor so data display starts at the top of the screen. When the screen is filled, three asterisks (***) display on the last line. Read the screen and press ENTER or CLEAR to erase the screen and continue the display of data.

The program function keys map into the functions shown in “Table 6-3” on page 6-15:

Table 6-3 VTAM Client Telnet Function Keys

Function Key	Associated Command	Action
PF1/PF13	**HELP**	Displays PF key usage.
PF2/PF14	n/a	No action.
PF3/PF15	**END**	Disconnects the terminal from TCPaccess.
PF4/PF16	<**AYT**>	Sends the Telnet **Are You There** message. Any operand is ignored.
PF5/PF17	<**AO**>	Sends the Telnet **Abort Output** message. Any operand is ignored.
PF6/PF18	<**BRK**>	Sends the Telnet **Break** message.
PF8/PF20	**XCTL**	Sends typed data, with the last character as **CNTL-*x***.
PF9/PF21	**HIDE**	Non-display of next input line.
PF10/PF22	**XWNL**	Sends typed data with **CRLF.**
PF11/PF23	**XNNL**	Sends typed data without **CRLF.**
PF12/PF24	**END**	Disconnects the terminal from TCPaccess.
PA1	**IP**	Sends the Telnet **Interrupt Process** message.
PA2	**KO**	Sends both **IP** then **AO**.

Only Enter and PF10/PF22 append CRLF to the data sent. PF7/PF19 appends the ESC character and PF8/PF20 takes the last character typed and converts it to a control character before sending. Enter and PF10/PF22 are similar, but Enter causes the cursor to move to the next line while PF10/PF22 sends a blank line without moving to the next line. Use Enter on SNA 3278 terminals, attention maps into the PA1 key for compatibility.

Full-Screen Operation from 3278 Terminals

If the remote server host is an IBM system (either MVS or VM) and the local user has a 3278-type terminal, **VTELNET** negotiates transparent full-screen mode with the server host.

In full-screen mode, all keys are transmitted and no local action is performed. If you must abruptly disconnect from TCPaccess while in full-screen mode, use the 3278 system request key (SYS REQ).

Client Telnet for ASCII Terminals

TCPaccess can support Client Telnet access from ASCII terminals and other non-3270 terminals that support remote echo mode. You can run Client Telnet from qualifying clients as if they were full-screen 3270 facilities. Among the supported terminal types are:

- DEC VT52
- DEC VT100
- DEC VT220
- DEC VT320
- IBM316x
- TeleVideo 905
- Zentec 8031
- AT&T 610
- Hewlett-Packard terminals that have TCP/IP capability

The above list is not all-inclusive. Other kinds of terminals can also process Client Telnet commands. You can display an online listing of terminals supported at your site. See "Invoking Client Telnet from an ASCII Terminal" on page 6-17 for details.

Note: Certain configuration steps are required to set up Client Telnet access for ASCII and other non-TN3270 terminals, including the installation of Simware Sim3278 TCP/IP software on the target host(s). For details, refer to the ***TCPaccess Customization Guide.***

Invoking Client Telnet from an ASCII Terminal

Follow these steps to invoke Client Telnet from an ASCII terminal or other non-3270 terminal:

1. At the system prompt, enter the **TELNET** command followed by the host name, and press ENTER. For example:

TELNET *my_host*

2. At the prompt, enter the command to call the service you want to access. The availability of commands and services is determined by site system managers. Consult your system manager to obtain the command(s) that are valid at your site. For example:

TSO

3. You are prompted to either enter the terminal type or log off.

   If you are not sure whether your terminal is supported, type a question mark (?) at the prompt to see a list of supported terminals for your site. For example:
   Otherwise, enter the terminal type and press ENTER.

VT220

4. The software validates the terminal access and connects you to the target service.

   You can use normal commands and operations to invoke features offered by that service, and to logoff the service.

Server Telnet

The Server Telnet facilities provide remote access to host application programs through the Telnet protocol. The Server Telnet facility accesses a server subsystem that drives a supported terminal type via ACF/VTAM. This includes TSO, CICS, and other popular subsystems.

The supported (virtual) terminal types are:

- IBM 3767 typewriter terminals (SNA LU1 virtual terminals)
- Locally connected non-SNA IBM 3278 terminals (SNA LU0 virtual terminals)

Either of these can be driven from an NVT to provide line-at-a-time operation for the remote user. The virtual 3278 can also be used in transparent full-screen mode. When you are a remote user from a remote IBM MVS or VM system and open a TCP connection to the well known Telnet server port (23), you are connected to a Telnet server process (ULPP) in TCPaccess. If you proceed to logon to TSO, for example, the Telnet server ULPP invokes the Virtual Terminal Facility (VTF), which uses ACF/VTAM to make a cross-address-space connection to the virtual terminal handler for TSO. The ULPP makes all necessary conversions of code and protocols.

Remote users on non-IBM hosts who want to connect to applications in full-screen mode must have 3270 client emulation software on their host.

Server Telnet also supports the use of Session Level USSTAB (Unformatted System Services Tables) and the associated msg10 screens, as described later in this chapter.

Server Telnet Commands

The Telnet server also implements these pre-logon services within TCPaccess:

- **HELP**

  Displays available commands.

- **HELP** *command*

  Displays help information for that command.

- **NEWS**

  Displays the news data set.

- **BYE, CLOSE, END, QUIT, or LOGOFF**

  Causes Server Telnet to close the connection.

- **NETSTAT**

  Provides status information regarding TCPaccess. For system programmers, an alternate entry called **SYSSTAT** is provided that enables the TCPaccess control functions in **NETSTAT**. The Telnet server requires a local **LOGON** before allowing access to **SYSSTAT**. The subcommands of **NETSTAT** are documented in the ***TCPaccess System Managment Guide***.

- **SIGNON or LOGIN**

  Prompts for user ID and password.

- **SIGNOFF or LOGOUT**

  Logs out the user.

- **ACTEST**

  This is the TCPaccess interactive debugging tool. It requires a local **LOGON** from the Server Telnet and is restricted to system programmers. The subcommands of **ACTEST** are documented in the ***TCPaccess Customization Guide***.

Autologon to Specific VTAM Applications

Many sites with session manager software prefer their users to be automatically connected to the session manager. This requires minor changes to the configuration file. Read the *TCPaccess Customization Guide* for details.

USS Table Support for Server Telnet

Server Telnet supports the use of Session Level USSTAB (Unformatted System Services Tables) and their associated msg10 screens. The feature enables you to customize screen access information for VTAM applications that are opened through TCPaccess.

For more information about USS table support, read the ***TCPaccess Customization Guide.***

Telnet Escape Sequences

TCPaccess implements the Telnet protocol using the logical not character (¬) as an escape character. This character must be doubled (¬¬) to be transmitted correctly.

An escape sequence is a single Telnet escape character ¬, followed by a predefined character sequence. The entire sequence represents a single character, usually nongraphic or one that is not available on all IBM keyboards.

Valid Escape Sequences

The sequences described in "Table 6-4" on page 6-20 are valid:

Table 6-4 Valid Escape Sequences

Sequence	Description
¬_x	*x* is any EBCDIC character. This represents the ASCII CONTROL-SHIFT of *x*. More precisely, it represents the low-order 5 bits of the ASCII equivalent to *x*.
¬=x	
or	
¬@X	*x* is any EBCDIC alphabetic. This represents the inverse alphabetic shift of *x*.
¬<	An ASCII left bracket.
¬>	An ASCII right bracket.
¬(	An ASCII left brace.
¬)	An ASCII right brace.
¬"	An ASCII caret.
¬'	An ASCII accent.
¬x	*x* is a valid control mnemonic string, in all one case. It can be one of the standard ASCII control mnemonics (LF, CR, FF, BEL, CAN, DEL, ETX, etc.) or one of these Telnet control mnemonics: AO DM EC IP SB AYT SO EL IAC WLL BRK DNT GA NOP WNT

7

Simple Mail Transfer Protocol

This chapter describes the Simple Mail Transfer Protocol (SMTP). It includes these sections:

- "Introducing SMTP" on page 7-2

  A brief description of the SMTP, including definitions of terms used in this chapter to describe SMTP.

- "Using the Mail Facilities" on page 7-5

  Describes how to use the TCPaccess mail facilities for the transportation of electronic mail.

- "Interface to a User Mail System" on page 7-8

  Describes the interfaces between the TCPaccess mail routines and the post office.

Introducing SMTP

The SMTP is used as the common mechanism for transporting electronic mail among different hosts within the Department of Defense Internet protocol suite.

Under SMTP, a user SMTP process opens a TCP connection to a server SMTP process on a remote host and attempts to send mail across the connection. The server SMTP listens for a TCP connection on a well-known port (25), and the user SMTP process initiates a connection on that port. When the TCP connection is successful, the two processes execute a simple request/response dialogue, defined by the SMTP protocol, in which the user process transmits the mail addresses of the originator and the recipient(s) for a message. When the server process accepts these mail addresses, the user process transmits the message. The message must contain a message header and message text formatted in accordance with **RFC 822**.

Mail Transport Support Programs

Within TCPaccess, these major programs support mail transport:

- SSMTP

  The SMTP Server program receives SMTP mail and spools it into a JES SYSOUT file.

- USMTP

  The SMTP User program assumes that mail to be sent is available in cataloged data sets with a specific DSNAME prefix. It awakens periodically (or in response to a signal from SPOOL#4 or SNDMSG) to try to send pending mail data sets.

- SPOOL#4

  The outgoing mail spooler copies output from a JES print queue into cataloged data sets that are the outgoing mail data sets for USMTP. SPOOL#4 then awakens USMTP to send the messages.

- SNDMSG

  This interactive program lets local users create outgoing mail files. Using data entered from a local user terminal, SNDMSG creates mail data sets in the same format as SPOOL#4. SNDMSG then awakens USMTP to send the messages.

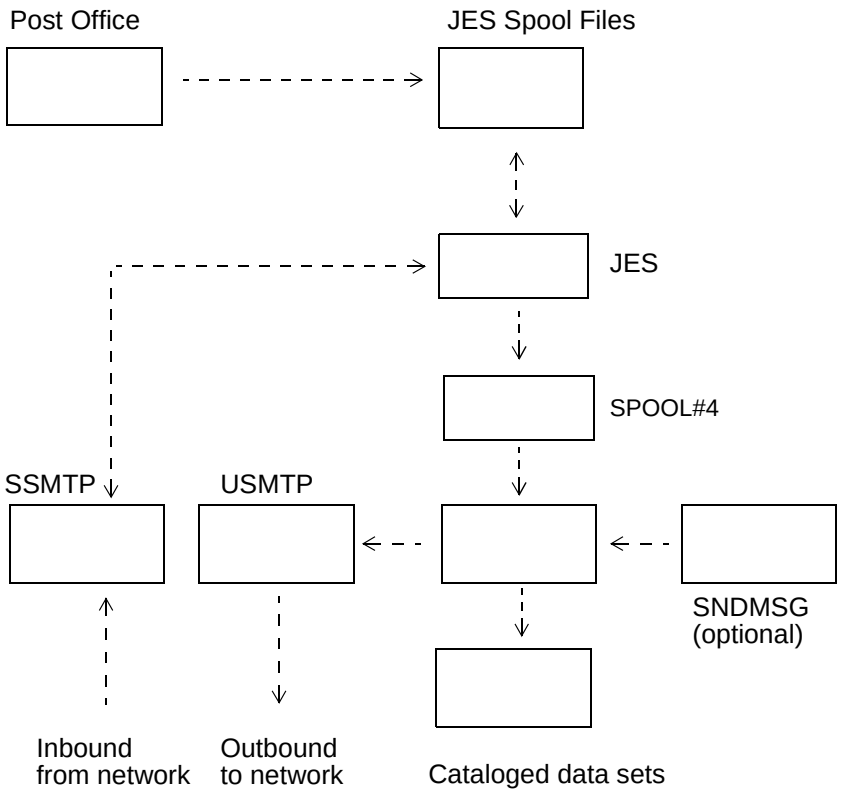
TCPaccess contains no mechanism to read the JES queue containing received mail (the queue written by SSMTP) nor to write the JES queue with mail to be transported (the queue read by SPOOL#4). To provide these mail services, an installation needs an appropriate user mail system. USMTP, SSMTP, and SPOOL#4 interface through JES2/NJE with a user mail system such as UCLAMAIL. However, since the interfaces are quite general, some other mail system can be substituted.

In this chapter, the user mail system is referred to as the post office, meaning either UCLAMAIL or some other mail system. Read “Interface to a User Mail System” on page 7-8 for a description of the interfaces between the TCPaccess mail routines and the post office.

Understanding TCPaccess Mail Services

"Figure 7-1" on page 7-3 shows the relationship of TCPaccess programs with one another and with the post office:

Figure 7-1 TCPaccess Mail Services



You can use the TCPaccess mail routines even without a post office. You can configure SSMTP to send received mail to a local printer for hard copy, and the SNDMSG program creates mail data sets for USMTP to send. Read "Using the Mail Facilities" on page 7-5 for information about direct use of the TCPaccess mail programs.

Mail Service Components

These terms are used throughout this chapter.

Mail Addresses	Mail addresses identify both the originator and recipient of a mail message. They generally take this form X@Y where X is a string, often a user or account, and Y is a string, often a host. In this chapter, X represents the local part of the mail address and Y represents the global part of the mail address. Mail addresses appear in the To: and From: fields of the message header and in the X-from: and X-to: fields of the envelope.
Envelope	The envelope is a header containing the originator and recipient mail addresses. It is prepended to each mail message by the post office, SNDMSG, or SSMTP, and it is used by all the TCPaccess SMTP routines as the message travels through the system. The envelope consists of the internally-defined (i.e., to the SMTP routines) header fields, X-from: and X-to: that let the SMTP routines convey mail addresses to one another.
Message Header	An **RFC 822** message consists of any number of header fields, optionally followed by message text. Typical header fields include: Date:, From:, To:, CC: (carbon copy), and Subject:. The **RFC 822** message header refers to the collection of these header fields.
Domain Literal	As defined in **RFC 822**, a domain literal is a dotted-decimal host number enclosed in square brackets. This is an example of a mail address using a domain literal: `CZQ14CD@[10.10.0.1]`
Multi-homed Host	Multi-homed hosts are connected to more than one network and therefore have multiple network addresses.
Host Names	These are mnemonic name strings by which hosts are known on the network. Each host has one official host name and can have optional nicknames. Although nicknames are allowed, their use is discouraged as discussed in **RFC 952**.
Domain Name Resolver	TCPaccess uses the Domain Name Resolver to map host names into the appropriate network addresses.

Using the Mail Facilities

This section describes how to use the TCPaccess mail facilities for the transportation of electronic mail.

Getting Started

Have your TCPaccess site administrator place an entry in the SMTP User Table for you. This table defines a mapping between TSO user IDs and user names. Each entry in the table contains these parameters:

- TSO user ID
- Mailbox name

  A name by which the TSO user is known at other hosts on your network. When SSMTP sends a message, it attempts to match the local part of the recipient mail address against this field to map it into a TSO user ID.

- Signature

  This optional parameter is used by USMTP in the creation of an originating mail address (the X-From: header field in the message header). For example, suppose the User Table had this entry:

  USMTP transforms the header field (created by either the post office or SNDMSG) from this format

```
X-From: CZQ14CD@MYHOST.ARPA
```

  into this format

```
X-From: <CDoaks@MYHOST.ARPA>
```

Receiving Mail

To receive mail, the SSMTP program can be configured to send mail to a local printer for hard copy output. The ***TCPaccess Customization Guide*** provides information on configuring SSMTP to send mail to the local printer.

Sending Mail

The TCPaccess SNDMSG program can interactively create mail data sets for USMTP to send.

Invoking SNDMSG

You can invoke SNDMSG directly from a local VTAM terminal or indirectly from a TSO TELNET session.

- To invoke SNDMSG directly from a local VTAM terminal, enter this command (in one of two forms) on the system logon invitation screen in place of the **LOGON** command:

ACCES ;SNDMSG

or

LOGON APPLID(ACCES) DATA(;SNDMSG)

Consult with your TCPaccess site administrator to find which of the two command forms to use.

- To invoke SNDMSG indirectly from a TSO TELNET session, enter the following command after the TELNET input prompt:

;SNDMSG

After the appropriate command is entered, the terminal is connected to TCPaccess and prompts you for a user ID and password:

```
REQUESTED SERVICE IS RESTRICTED
PLEASE ENTER USERID:
AND PASSWORD:
```

Consult with your TCPaccess site administrator for password and ID information. After permission is granted, control is passed to SNDMSG.

SNDMSG Session

SNDMSG first displays a herald message and then prompts you for recipient mail address(es), carbon copy mail address(es), subject, and message text. A sample SNDMSG session is shown here.

Using the data you entered, SNDMSG creates a mail data set containing an envelope, a message header, and message text in the same format that SPOOL#4 would create. Read “SPOOL#4” on page 7-9 for more information.

```
**SNDMSG ** WED, 13 JAN 91 16:40:47 EST
To:  abc@unix.md.company.com,xyz@unix.md.company.com.
CC:  mno@unix.md.company.com
Subject:  new test
Enter message text, followed by empty line. Use \x to
cancel last line, or \t to display msg. .
.
message text .
.
-->abc@unix.md.company.com SPOOLED
-->xyz@unix.md.company.com SPOOLED
-->mno@unix.md.company.com SPOOLED
```

Note: To send an empty line, press the function key PF10 (XWNL) to transmit with a null line.

Usage Guidelines

- You can enter multiple recipient or carbon copy mail addresses (**To:** or **CC:** address fields), separated by commas. SNDMSG creates a separate mail data set for each recipient and carbon copy mail address. Each contains a single **X-to:** field in the envelope.
- File inclusion is a message line beginning with the three characters ..f= followed by a data set name that causes the contents of that data set to be included in the outgoing message.

  Do not enclose the data set name in quotes; it must be fully qualified. This is an example:

```
.f=OTAPXYZ.MYMESSGE.TEXT
```

  The user and the TCPaccess user ID must have access authority to read the file. Also, if the file is not in the catalog, it is not sent.

  The file is expanded into the mail message immediately. This may affect virtual storage requirements if the file is large.

SNDMSG Termination

SNDMSG terminates automatically when you have completed entering your message. If SNDMSG is invoked directly from VTAM, control returns to VTAM and the terminal again displays the login invitation. If SNDMSG is invoked from a TSO TELNET session, control returns there.

Interface to a User Mail System

This section describes the interfaces between the TCPaccess mail routines and the post office in more detail.

Receiving Mail

The post office receives a message from the network by retrieving it from a JES SYSOUT file where it was previously spooled by SSMTP (described in the following section). The post office can retrieve these JES SYSOUT files using one of the following:

- The MVS-provided or a user-written external writer
- The JES Sub-System Interface (SSI)

SSMTP

SSMTP begins execution when it receives a request to connect to the well-known mail port. SSMTP completes the opening of the TCP connection and performs the receiver end of the SMTP protocol. When a message is received, SSMTP spools it into a JES SYSOUT file from which the post office retrieves it. A secondary copy of the message text is also spooled into a SYSOUT file destined for the local printer. Normally, this copy is deleted at deallocation time. However, if the received message exceeds a configuration-specified size, the primary copy (destined for the post office) is deleted and the secondary copy queued for printing. This is called diversion of the mail to the printer. (In the case where no post office is implemented, all received mail is diverted to the printer.) Both of the JES SYSOUT files have RECFM=VB (i.e., no carriage control).

SSMTP prepends an envelope to the messages it spools to JES. The envelope consists of a single X-from: field of the form:

X-from: remote user@remote host

followed by one or more X-to: fields of the form:

X-to: TSO userid@local post office name

The X-from: field contents are taken directly from the SMTP MAIL FROM: command. The X-to: field contents are derived from a single RCPT TO: SMTP command. The local part of the mail address from the RCPT TO: command is mapped into a TSO user ID using the SMTP User Table. If the local part of the mail address is not in the SMTP User Table, SSMTP rejects the recipient mail address. The local post office name is an SSMTP configuration parameter that represents the name of the local post office. It need not be the same as the local network host name.

Usage Guidelines

- The MAIL FROM: command received by SSMTP generates the X-FROM: line. The RCPT TO: commands received by SSMTP generate the X-TO: lines. The rest of the text of the file is placed into the file as received after the **DATA** command.
- ASCII format effectors in the input stream are generally handled as follows:
  - Horizontal tabs (HT) are expanded to blanks assuming tab stops in every 8th column: 8, 16, 24, 32, etc.

  - Other format effectors (FF, a LF not following a CR, a CR not preceding a LF>, a BS) appear in the spooled file as the corresponding EBCDIC control characters.

- SSMTP supports the following SMTP commands: **HELP O, MAIL, RCPT, DATA, VRFY, QUIT, RSET, NOOP, HELP**. This is the minimum implementation set.
- SSMTP does not support expanding mailing lists.

Example

The following example shows a sample incoming mail file as created by **SSMTP**:

```
X-FROM: abcdef@remotehost
X-TO: CZQ14CD@local post office name
To: CDoaks@OURHOST.ARPA
Date: Tue, 12 Jan 91 10:11:12 EST
From: abcdef@remotehost
Subject: Status of SMTP Documentation
What is the current status?
```

Sending Mail

The post office originates a message destined for your network by formatting it with an envelope, message header, and message text and then spooling the message to a JES queue.

The MAIL FROM: command sent by USMTP comes from the X-FROM: line. The RCPT TO: commands sent by USMTP comes from the X-TO: lines. The rest of the text of the file is sent by USMTP after the **DATA** command.

Example

This example shows a sample outgoing mail file:

```
X-FROM: CZQ14CD@OURHOST.ARPA
X-TO: abcdef@remotehost
To: abcdef@remotehost
Date: Tue, 12 Jan 91 10:11:12 EST
From: CZQ14CD@OURHOST.ARPA
Subject: Status of SMTP Documentation
The documentation has been successfully completed
```

SPOOL#4

SPOOL#4 uses the JES subsystem interface. SPOOL#4 reads messages from a JES queue and creates cataloged data sets. SPOOL#4 signals USMTP. The data sets created by either have RECFM=VBA and data set names in this form:

‘**directory.D***jjjjj***.T***hhmmss***0.N***xxxx***’**

Syntax Description

jjjjj	Julian date
hhmmss	Time of day in hours, minutes, and seconds
xxxx	The nth data set processed during that second

Note: **SNDMSG** creates data sets using this format:
'directory.D*jjjjj*.T*hhmmss*0'

The data set name prefix directory is a SPOOL#4/USMTP configuration parameter (the PATH parameter on the SMTP statement in the file APPCFGxx).

Sample Jobs

The following sample jobs can be used to send a mail file to RMT21 (the REMOTE parameter on statement SMTP in the APPCFGxx member) for SPOOL#4 to pick up and convert into a mail data set:

This first sample reads from a file (T01TCP.ACCES.EMAIL(NEWMAILD):

```
//ICSGENER JOB (TSO00,,,99),'WRITE TO SPOOL#4',
//MSGCLASS=X,CLASS=A
//*
//STEP1    EXEC PGM=IEBGENER
//SYSUT1 DD DISP=SHR,DSN=T01TCP.ACCES.EMAIL(NEWMAILD)
//SYSUT2 DD SYSOUT=A,DCB=*.SYSUT1,
//          DEST=RMT21
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//
```

Or, you may use the following to include the mail message in your job:

```
//ICSGENRM JOB (TSO00,,,999),'WRITE TO SPOOL#4',
//MSGCLASS=Z,CLASS=A
//*
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD *
 X-From: t01tcp@host1.HQ.company.com
 x-to: t01tcp@host2.md.company.com
 Date: Tue, 7 Nov 89 11:08:49 EST
 From: t01tcp@host1.md.company.com
 to: t01tcp@host2.md.company.com
 Subject: test mail

 We test mail through iebgener batch job.
 test mail  test mail  test mail  test mail  test mail  test mail
 This is the last line.
/*
//SYSUT2 DD SYSOUT=A,DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB),
//          DEST=RMT21
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
//
```

USMTP

USMTP awakens periodically (or when signaled by SPOOL#4 or SNDMSG) to try to send pending mail data sets. It tries to send each data set that is either new or whose retry timeout has expired. Any data sets successfully sent are deleted and uncataloged. The others are kept for later attempts. USMTP expects the mail data sets to be in the format shown in the example in the section on "Sending Mail" on page 7-9. USMTP assumes only a single X-TO: line (i.e., recipient mail address) per data set. For multiple network recipients, the post office (or SNDMSG) must create multiple data sets.

USMTP generally handles arbitrary carriage control characters in the mail data set translating them to corresponding sequences of ASCII format effectors.

If transmission of a particular message is unsuccessful because of a temporary error (e.g., the remote host is down), USMTP retries periodically for a limited number of times. If this number is exceeded, or there is a permanent error (e.g., the remote SMTP receiver refused the message because the target local part of the mail address was unknown), USMTP redelivers the message to the local user. For this redelivery, USMTP issues a message from the Mailer Daemon saying that the message was undeliverable and why. USMTP appends the Daemon message to the original message and spools it into the JES queue used by SSMTP (see the section of this chapter describing SSMTP) to deliver messages to the post office.

USMTP configuration parameters determine the frequency of retries and the maximum retry time.

USMTP has the following general features:

- Multi-homed Host Logic

  To provide robust delivery, USMTP tries each address in turn of a multi-homed host before deciding that the mail cannot be delivered.

- Host Name Mapping

  USMTP scans the fields in the message header and envelope that contain host names and, if necessary, maps nicknames into official host names.

- MX record support

  USMTP uses this feature to resolve mail routes for a host. (similar to a DNRGET ROUTE BYNAME host_name).

- TSO User ID Mapping

  As mentioned, USMTP scans the mail addresses in the envelope and message header and, if possible, uses the SMTP User Table to map local TSO user IDs into a mailbox name string before sending the message.

  Note: To use Host name mapping and MX record support, **USMTP** must be able to parse RFC 822 message headers.

- File Inclusion

  A message line beginning with the three characters ..f= followed by a data set name causes USMTP to include the contents of that data set in the outgoing message. Do not enclose the data set name in quotes; it must be fully qualified. This is an example:

```
.f=OTAPXYZ.MYMESSGE.TEXT
```

  The user and the TCP user ID must have access authority to read the file. If the file is not in the catalog, it is not sent.

Note: There is a potential access-control problem with this facility. Since USMTP does not have a password for the given user account, it cannot do a secondary login to assume the user's authority by any access control mechanism. Consequently, you must ensure that TCPaccess has read-access to the specified data set. Otherwise, the file is not sent.

Because included files are now expanded by SNDMSG, USMTP normally does not see ..f= commands. Such commands may have been entered by some other mailer via SPOOL#4.

- Support for Domain Literals

  You can send a message using a domain literal in place of a host name. However, since it is difficult to enter square brackets on an IBM system, USMTP accepts IP address domain literals without brackets:

CZQ14CD@10.1.0.1

8

USPOOL

This chapter describes the print server program, USPOOL for the UNIX **lpr** command. It contains these sections:

- "Overview of the Print Server Program" on page 8-2

  Provides a brief overview of the print server program, USPOOL program and the commands it uses.

- "UNIX Print Facility" on page 8-3

  Describes the UNIX lpr facility and the lp commands.

- "Protocol and Format" on page 8-4

  Describes the protocol and format the lp commands need.

- "Print Server Facility" on page 8-8

  Describes the TCPaccess print server facility, USPOOL.

Overview of the Print Server Program

The print server program, USPOOL, runs as a server process within TCPaccess. USPOOL lets a UNIX user route print data through a network and print it on a line printer that is attached to the IBM host. Print requests are generated on the UNIX system with the UNIX lpr command.

The formats and protocols used by USPOOL and the lpr facility are described in this chapter. Information required by UNIX system programmers to make the necessary resource definitions is also included. Information required by IBM system programmers is mentioned here and described in the ***TCPaccess Customization Guide***. Users with an interest in a more complete lpr server and lpr client should investigate the Enterprise Print Services product.

While the information in this chapter refers to UNIX-based systems, any system implementing the USPOOL lpr formats and protocols can use the USPOOL print server program. The lpr command has been ported to some IBM PC-based TCP/IP products.

UNIX Print Facility

The UNIX **lpr** command initiates print requests from a UNIX system. If the print request is destined to a remote printer, the remote printer daemon opens a TCP connection (default port 515) to the host that owns the remote printer. It then transfers the print data to the remote host using the prescribed formats and protocols.

The UNIX lp Commands

In addition to the lpr command, other lp commands require processing at the remote host. These commands also operate across a TCP connection to remote port 515. These sections describe the lp commands. See the UNIX User's Manual for more information about these UNIX commands.

lpr

The UNIX **lpr** command creates remote and local print requests. For remote requests, the printer daemon transmits two files across the network to the remote system. One consists of the file to be printed, the other consists of control information such as print classification, job name, and title. USPOOL currently reads the control file, but does not process its contents. The data file is printed using standard defaults.

lpq

The UNIX **lpq** command queries the status of pending print requests, both local and remote. USPOOL ignores this command, and no response is printed at your terminal.

lprm

The UNIX **lprm** command `cancels` pending print requests generated by the **lpr** command. This applies to both local and remote printers. USPOOL support for remote **lprm** processing is not yet implemented; when issued to TCPaccess, this command is ignored.

Line Printer Daemon

The line printer daemon is the UNIX server that initiates the operations of **lp** requests. The daemon listens for TCP connections on default port 515. Once a connection has been established, the remote system sends the character string indicating the type of **lp** request and the printer name. The daemon then initiates the desired action.

Remote Printer Definition

A remote printer is defined to a UNIX (BSD 4.3) system in the file /etc/printcap. This file assigns the remote printer and remote host names as well as other printer configuration information. The remote printer name assigned here is transferred over the TCP connection when an **lp** request is made to a remote host. Consult your UNIX system administrator or appropriate UNIX manual to define the remote printer.

Protocol and Format

Once a TCP connection to port 515 has been established by the line printer daemon, lp commands and data can be exchanged between the two systems. First, the UNIX system sends a character string to the remote computer to identify the type of request (lpr, lpq, or lprm) and the name of the remote printer the request is for. An lpr request causes print files and control information to be sent across the TCP connection as described in following sections. The lpq command causes status information to be returned to the requester. The lprm command causes the result of the requested operation to be returned.

Overview of the lpr Protocol

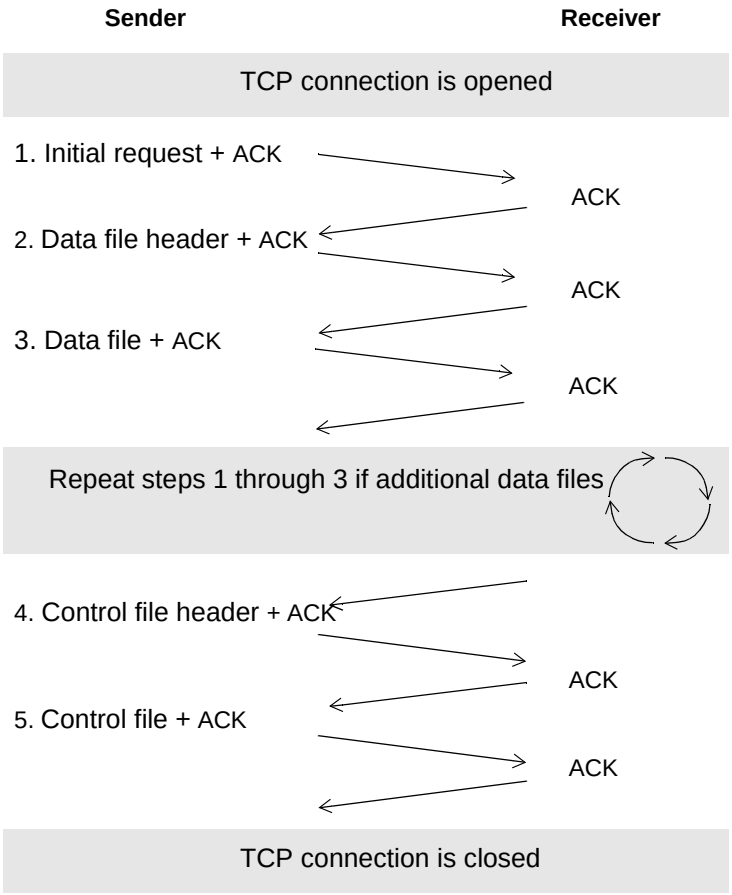
This section describes the sequence of events that effects the sending of lpr print data from the local to the remote computer. (See “Figure 8-1” on page 8-4.)This protocol is implemented over a TCP connection. Acknowledgments (ACKs) delimit the phases of the transfer. In **lpr**, an ACK is one zero value byte. There are three phases of an lpr transfer.

- Initial request
- Transfer of one or more data files to be printed
  - Transfer of the file header
  - Transfer of the file data
- Transfer of a control file
  - Transfer of the file header
  - Transfer of the file data

For the first phase (the transfer of the file header) the sender transfers a character string that identifies the type of request and the printer name. The first byte of the character string represents the request type. It has one of the values listed in “Table 8-1” on page 8-6:

The ASCII character string that immediately follows the request type character is the remote printer name destination. The remote printer name is terminated with one zero value byte (ACK). The format of the data and control files is described in the following sections.

Figure 8-1 Flow of a Normal lpr Transaction



Data File Format

The data file consists of a data file header followed by the print data.

The data file header has the format

\3*XXXXX dfYZZZAAAAA***\n**

Syntax Description

\3	Indicates the data file
XXXXX	Number of characters in the data file
df	specifies the data file
Y	File sequence associated with job number
ZZZ	Job number
AAAAA	Sending host name
\n	End of data file header

Usage Guidelines

- The strings *XXXXX* and *AAAAA* are variable-length fields.
- The string *d fYZZZAAAAA* is the data file name. All characters are ASCII.
- The remote computer must send an acknowledgment of the data file header before the data file is sent.
- The data file consists of only the number of ASCII characters indicated in the data file header. The data file is terminated with 0 (an ACK).
- The remote computer must again send an ACK when receipt of the data file (including the \0) is complete.
- The sender then transmits either the next data file (if any), or the control file. Data files should not be printed at this point, but should be spooled temporarily so the control commands can manipulate them as directed by the originating computer.
- The current USPOOL implementation does not spool the file at this point, but immediately queues the file for printing. Printing of the data files by USPOOL is currently not affected by the contents of the control file.

Control File Format

The control file is similar to the data file in that they both consist of a header followed by the contents of the file. In fact, the control file header is identical to the data file header except for the file-type designator (2 for the control files) and the first character of the file name (c for control file). Acknowledgment of the control file and control file header is the same as for data files.

The control file consists of a series of commands describing actions and options for processing the previously spooled data files. The command verb is the first character of each line in the command file. Arguments make up the remainder of the line.

"Table 8-1" on page 8-6 lists the control file commands:

Table 8-1 Control File Format

Command	Description
J	Places the job name on banner page
C	Places the class name on banner page
L	Prints literal user's name on banner
T	Provides a title for the print file
H	Indicates the name of host machine where the lpr command was issued
P	Indicates the user's login name
I	Specifies the amount of space to indent the output
f	Specifies the name of the text file to print
l	Specifies the name of the text file with control characters
p	Specifies the name of the text file to print with pr (1)
t	Specifies the name of the troff (1) file to print
n	Specifies the name of the ditroff (1) file to print
d	Specifies the name of the dvi file to print
g	Specifies the name of the plot (1G) file to print
v	Specifies the name of the plain raster file to print
c	Specifies the name of the c if plot file to print
1	Specifies the R font file for troff
1	Specifies the B font file for troff
4	Specifies the S font file for troff
N	Specifies the name of file (used by lpq)
U	Specifies the unlink name of file to remove
M	Mails to user when done printing

Print Server Facility

The TCPaccess print server facility provides a print server capability that lets remote users access MVS printers.

USPOOL Overview

USPOOL is invoked by the incoming logger function of TCPaccess when a connection request (TCSYN) is sent to port 515 (or whatever port number the UNIX and IBM system programmers have defined). USPOOL initialization consists of obtaining a reentrant work area from TCPaccess, locating USPOOL SERVICE APPCFGxx parameters, completing the TCP connection open, validating the remote host, loading the translate table, initializing buffers, processing the lp request string, and allocating and opening the SYSOUT data set. USPOOL then drops into the main loop. Each iteration of the main loop handles a data file or the control file and send required acknowledgments. In the case of data files, an inner loop invokes a hierarchy of programs to get data from the network, format print records, translate ASCII to EBCDIC, handle the ASCII control characters (BS, HT, LF, FF, CR), write the records to the JES SYSOUT queue, and check for and process acknowledgments received.

When all functions have been completed, USPOOL does a program cleanup and closes the TCP connection.

ASCII-EBCDIC Translation

You can replace the standard **USPOOL** translate table with your own. Read the ***TCPaccess Customization Guide*** for details. If you have a 3800 printer you can use a replacement translate table to reproduce the ASCII characters.

Control Character Translation

USPOOL supports the ASCII backspace (x'08'), horizontal tab (x'09'), line feed (x'0A'), form feed (x'0C'), and carriage return (x'0D') control characters. These characters, except horizontal tab, are translated into the appropriate IBM machine code control character (the first character on the print line) in print-then-act mode. Read the control characters appendix in your most current version of the IBM publication, ***MVS/370 Data Management Services***, for descriptions of the machine codes.

Backspacing

Since it is impossible to do character backspacing on a line printer, a line that contains backspace characters must be formatted into multiple print lines. These print lines must then be printed with the *print-only* machine code set in all but the last line.

Horizontal Tab

A horizontal tab character causes the print line to be padded with blanks up to the next module 8 print position.

Line Feed

The current line is printed and formatting begins for the next print line.

Form Feed

The Skip to Channel 1 machine code is set in the current line. The line is printed and formatting begins for the next line.

Carriage Return

The print only machine code is set in the current line. The line is printed and formatting begins for the next line.

Limitations and Restrictions

TCPaccess does not support the **lpq** or **lprm** commands.

Although these two limitations exist in TCPaccess, the Enterprise Print Services provides support for them:

- TCPaccess does process the print control file, because data files are not spooled but immediately queued for printing. EPS does process the print control file.
- TCPaccess does not provide an operator interface for printing, but EPS does.

USPOOL Configuration

Parameter support for the USPOOL LPR statement in APPCFGxx lets an installation select the SYSOUT class of print requests, the name of a module used as the translate table, the size (number of bytes) used for the TCP buffers, and whether all print requests, only print requests from the local network, or only print requests from the local subnet are accepted for printing. You can change the TCP port from the default (515), but you must change this on the UNIX system also. Read the ***TCPaccess Customization Guide*** for information about how to manipulate the APPCFGxx parameters.

9

Remote Executor

This chapter describes how to use the TCPaccess Remote Executor. It includes these sections:

- "What Is the Remote Executor?" on page 9-2

  A brief description of the Remote Executor, TCPREXEC.

- "The REMCMND Program" on page 9-2

  Describes how the REMCMND program passes information to the REXEC daemon.

- "Running REMCMND Interactively (TCPREXEC)" on page 9-3

  Describes how to run the REMCMND program from a TSO command line with the TCPREXEC command.

- "Running REMCMND In Batch Mode" on page 9-5

  Describes how to run the remote executor in batch mode. Example JCL is provided.

What Is the Remote Executor?

The Remote Executor, TCPREXEC, uses the REMCMND program to execute a command on a foreign host and receive results on the local host. You can use the TSO command TCPREXEC or you can run TCPREXEC in batch mode using a JCL.

This product uses version 6.0 of SAS C.

The REMCMND Program

The REMCMND program lets you execute a command on a remote host and receive results on the local host. The REMCMND program can be executed from the TSO command line or from a batch job using a JCL.

An REXEC daemon must be running on the host. The REMCMND client passes the user name, password, and command to the REXEC daemon. The daemon provides logon and user authentication, depending on the parameters set by the user. If authentication fails, an error message displays on the local host.

Running REMCMND Interactively (TCPREXEC)

When executing REMCMND interactively under TSO, you must invoke a REXX EXEC called TCPREXEC to dynamically allocate statements and files. TCPREXEC passes command-line arguments to the host in as-is format (overriding the TSO function which makes all arguments uppercase). The TCPREXEC REXX EXEC is contained in the SAMP library. Copy it to a system CLIST or REXX library to use it.

Note: Information can be passed to **REMCMND** by either the **PARM** string or the **SYSIN DD** file. However, data cannot be entered both on the **PARM** string and in the **SYSIN DD** file.

To execute a command on the foreign host and receive the results on the local host, use the **TCPREXEC** command as shown here:

TCPREXEC **'{USER** (*username*)}
{PASS (*password*)}
[{SUBSYS (*subsysid*)}]
hostname
*command_string***'**

Syntax Description

USER	Specifies the user ID on the foreign host. The default user name (if you wait for the prompt) is your TSO user ID, in upper case. If your user ID is the same on the host system but in lower case, it must be retyped (in lower case). If this parameter is not supplied, you will be prompted for it.
PASS	Specifies the password of the user ID on the foreign host. Do not specify PASS() on the command line if no password is used. If prompted for a password, just press the enter key. If this parameter is not supplied, you will be prompted for it.
SUBSYS	Specifies a particular TCPaccess gateway rather than the default ACSS. This parameter is optional.
hostname	The name of the foreign host that is the target of the **REMCMND** command. hostname can be a fully qualified name or alias, or the IP net address, in the form nnn.nnn.nnn.nnn. This parameter is required.
command_string	The command sent to the foreign host, composed of one or more words. The code checks for the user name, password, subsystem ID, and host name, then assigns the remaining string as the command.

Using the NETRC File with TCPREXEC

REMCMND uses a NETRC (user.FTP.NETRC on MVS) file if you provide one. The NETRC file contains the host name, user ID, and password for multiple machines on your network. It takes this format:

MACHINE *hostname*

LOGIN *username*

PASSWORD *password*

Use this sequence of MACHINE, LOGIN, PASSWORD for each of the hosts you want to connect to.

TSO prepends the current local user ID to FTP.NETRC, then searches the MVS catalog for that file. When TCPREXEC finds user.FTP.NETRC, it reads that data set, looking for the machine name you requested (hostname). When a match is found TCPREXEC reads the login and password statements and uses those for authentication to the remote host.

Note: When a NETRC file is used, the case of the machine name, login name, and password must match the case of the name used in the **SYSIN**.

Example:

To have the **TCPREXEC** program scan your NETRC file for a machine with the host name snoopy, enter this **TCPREXEC** command:

TCPREXEC snoopy ls -l

In this command, snoopy is the host name and **ls -l** is the command string.

If TCPREXEC does not find user.FTP.NETRC, or does not find the host name in the file, it prompts for the user name and password.

Note: FTP2 uses the same NETRC file that TCPREXEC uses. Read "Using the NETRC File with TCPREXEC" on page 9-4 for more information.

Running REMCMND In Batch Mode

To run REMCMND in batch mode, you must include information in a JCL stream.

If you want to run in batch mode, use the JCL in this example. This example JCL issues an **ls -l** command on the remote host *hostname*.

```
//REMCMND    JOB
//JS01 EXEC PGM=REMCMND,
//  PARM='USER(username) PASS(password) hostname ls -l'
//STEPLIB    DD DISP=SHR,DSN=T01TCP.LINK
//           DD DISP=SHR,DSN=T01TCP.LOAD
//SYSPRINT   DD SYSOUT=A
//SYSTERM    DD  SYSOUT=A
//OUTPUT     DD SYSOUT=A
```

Output received from the command executed by program REMCMND is written to the OUTPUT DD statement.

Using a NETRC file in Batch Mode

When you run REMCMND in batch mode, REMCMND does not automatically read your NETRC file as it does for the interactive TSO command TCPREXEC. In batch mode, you must include a NETRC statement in your JCL.

The next example uses the user.FTP.NETRC file to provide the user name and password.

```
//REMCMND     JOB
//JS01 EXEC PGM=REMCMND,PARM='hostname ls -l'
//STEPLIB     DD DISP=SHR,DSN=T01TCP.LINK
//            DD DISP=SHR,DSN=T01TCP.LOAD
//NETRC       DD DSN=uid.FTP.NETRC,DISP=SHR
//SYSPRINT    DD SYSOUT=A
//SYSTERM     DD  SYSOUT=A
//OUTPUT      DD SYSOUT=A
```

Note: FTP2 uses the userid.FTP.NETRC file. Both FTP2 and REMCMND can use the same NETRC file. Read "Client FTP2" on page 2-4 in "Client FTP2" and "Using the NETRC File with TCPREXEC" on page 9-4 for more information about the NETRC file.

Command Line Arguments

You can use the PARM option on the EXEC statement to pass information to the REMCMND program.

```
//JS01 EXEC PGM=REMCMND,PARM='host_name cmnd_string'
```

Combining Commands

On many systems, you can string several commands together on the same line. For example, on UNIX, you can issue two commands on the same line by separating the commands with the semicolon (;) as a delimiter.

Examples

% ls ; date

You can use this facility with REMCMND in batch files.

```
//JS01 EXEC PGM=REMCMND,PARM='hostname ls;date'
```

Note: Because the TSO parser uses the semicolon (;) delimiter for parsing TSO commands, you cannot embed the semicolon in the command string when using the **TCPREXEC** TSO command.

Using a SYSIN DD Statement In Your JCL

If the parameter field on the EXEC statement is blank, REMCMND looks for a SYSIN DD statement in the batch JCL. REMCMND reads an 80 byte card from SYSIN and parses it like the command-line arguments.

Note: The SYSIN file is read only if the job is running in batch mode.

Include information for *username, password, hostname,* and *command_string* in the JCL as shown in this example:

```
//REMCMND   JOB
//JS01 EXEC PGM=REMCMND
//STEPLIB   DD DISP=SHR,DSN=T01TCP.LINK
//          DD DISP=SHR,DSN=T01TCP.LOAD
//SYSPRINT  DD SYSOUT=A
//SYSTERM   DD  SYSOUT=A
//OUTPUT    DD SYSOUT=A
//SYSIN     DD *
user_name password host_name cmnd_string
//
```

Note: Since your parameters can be specified in the **SYSIN** file, the **PARM** field on the **EXEC** statement is not needed.

For frequently used commands, you can include this line in the JCL to create a control file with the pertinent information:

```
USER(user_name) PASS(password) zulu exec /usr/openwin/bin/xioctetn \
-server telnet -host mvshost -title REMCMND
```

Using the \ Character for Continuing Long Lines

The backslash character (\) is used to continue a line of text that does not fit on the screen. The backslash may appear anywhere on the line but must be followed by at least one blank space.

CAUTION ! Due to parsing limitations, the input statement specified via SYSIN must ***not*** contain sequence numbers in columns 72 through 80, since it will be sent as data to the remote host and cause an error.

10

Using TCPaccess API and Socket Applications

This chapter describes some API and socket application programs that you might find useful. It includes these sections:

- "What Example Applications are Provided?" on page 10-2

  Briefly describes how these examples are presented and their uses.

- "ACSHELLO" on page 10-3

  This TSO command acts as a server that can be queried by clients to obtain information about the host on which the ACSHELLO program is executing.

- "FINGER" on page 10-4

  This TSO command obtains information about users of a system.

- "TTCP" on page 10-5

  This TSO command acts as a data source or sink facility.

- "WHOIS" on page 10-8

  This TSO command queries a default or user-specified WHOIS server to obtain information about a user-specified name.

What Example Applications are Provided?

The FINGER, WHOIS, ACSHELLO, and TTCP application programs are provided in load module and source forms to illustrate use of the socket and basic C libraries.

The ACSHELLO and WHOIS programs distributed as executable programs were created form the BHELLO and BWHOIS sample code.

The TTCP program is provided in load module and source forms as an example of how to use the assembler API macro facilities.

Any errors encountered by the programs described in this chapter are indicated to the user by error messages. For information about these error messages, see ***TCPaccess Prefixed Messages and Codes*** and ***TCPaccess Unprefixed Messages***.

These programs might prove valuable to network users. For more information on additional diagnostic commands, see the ***TCPaccess Customization Guide***. These programs are invoked through TSO or a batch job. Instructions for using each program are provided in the following subsections.

ACSHELLO

The **ACSHELLO** program acts as a server that can be queried by clients to obtain information about the host on which the **ACSHELLO** program is executing. This is a nonstandard network service but might be useful to users as a demonstration of the network capabilities. It also provides a good example of the implementation of a network server. The program supports access only via TCP. This program runs until canceled or interrupted, or until an unrecoverable error is detected.

ACSHELLO [**PORT(** *TCP_port* **)**] [**SYSID(** *subsystem_ID* **)**]

Syntax Description

ACSHELLO	Invokes the **ACSHELLO** program.
PORT(*TCP_port*)	Defines the port the program will use to wait for client requests
SYSID(*subsystem_ID*)	Lets the user specify the subsystem ID of TCPaccess other than the default used when TCPaccess was installed.

Usage Notes

- The default port is 43; it is typically used as the WHOIS TCP service.
- The default subsystem ID is ACSS.
- Both the PORT and SYSID parameters are optional.

Example

The following is sample output of the ACSHELLO program when contacted by a client:

```
ACSHELLO:  Server starting at:            Wed Feb 07 13:57:27 1990
ACSHELLO:  Running on host:               ZEUS.MD.INTERLINK.COM
ACSHELLO:  Host's internet address is:    129.192.192.129
ACSHELLO:  MVS version level is:          MVS/SP2.2.0
ACSHELLO:  MVS SMF ID is:                 IPO1
ACSHELLO:  CPU ID is:                     000130614381
ACSHELLO:  Server stopping at:            Wed Feb 07 13:57:28 1990
```

Invoking ACSHELLO

Typically this program is used by starting its execution on the mainframe and then querying it using TELNET or WHOIS, specifying the port that ACSHELLO is using to listen for requests.

If you are using the WHOIS program to query the ACSHELLO server using the default port, invoke the program with this command:

WHOIS NAME(*ABC*) **HOST**(*my_host_name*)

FINGER

The FINGER TSO command processor is similar to the UNIX finger command. It obtains information about users of a system. If given only the name of the remote host, FINGER returns information about all users currently logged in on that system. If given the name of a user and the remote host, FINGER returns information about that user. The finger protocol is described in detail in RFC 742.

FINGER NAME(*lookup_name*) **[LONG] [SYSID**(*subsystem_ID*) **]**

Syntax Description

FINGER	is the command used to obtain information about users logged in to a remote host
NAME(*lookup_name*)	specifies the user name for which information is requested
LONG	provides a more verbose listing of the remote system users' information if the *lookup_name* specifies only a remote host
SYSID(*subsystem_ID*)	allows you to specify the subsystem ID of TCPaccess other than the default used when TCPaccess was installed

Usage Notes:

- The remote host must be running a finger server program for the client to function properly.
- To obtain information about a single user, lookup_name should be a string in the form of userid @hostname.
- To obtain information about all users currently logged on to the host, type @hostname.
- The user ID is converted to lowercase before it is sent to the remote system. The finger server on the remote system should be case insensitive.
- The NAME(lookup_name) parameter is required.
- The LONG parameter is optional.
- The default subsystem ID is ACSS.
- The SYSID(subsystem_ID) parameter is optional.

Example

The following is sample output of the **FINGER** program:

```
FINGER NAME(gaw@leo)
(LEO.MD.INTERLINK.COM.)
Login name: gaw         In real life:..Glen......Directory:/home/gaw Shell:
/bin/csh
On since Feb 28 14:53:11 on ttyp2              6 hours 19 minutes Idle Time
```

TTCP

The **TTCP** TSO command processor acts as a data source or sink facility. TTCP is basically a test program for exercising TCPaccess and API functions. TTCP is compatible with the TTCP program available through anonymous FTP for UNIX users.

When run in transmit mode, TTCP performs one iteration and exits. When run in receive mode (as a server), TTCP runs until stopped. To stop TTCP when under TSO, use the particular TSO attention key for your terminal. To stop TTCP when running under the TSO TMP in a batch job, use the MVS **STOP(P)** command.

Any errors encountered by the TTCP program are indicated by error messages. For information about these error messages see *TCPaccess Prefixed Messages and Codes*.

The syntax for the **TTCP** command is as follows:

TTCP[RECEIVE/TRANSMIT] [TCP/UDP] [ASCII/EBCDIC]

[**HOST**(*host_name*)] [**PORT**(*port_number*)] [**SYSID**(*subsystem_ID*)]

[**BUFLEN**(*buffer_length*)] [**BUFNUM**(*number_of_buffers*)]

[**COUNT**(*transmit_buffer_count*)] [**TASKS**(*number_of_receive_tasks*)]

Syntax Description:

TTCP	Name of the command to be executed
RECEIVE	Indicates that TTCP should run as a server to receive and discard (sink) all data from any host and port; RECEIVE may also be specified as RECV or RCV
TRANSMIT	Indicates that TTCP should run as a client to generate (source) and transmit data to a specified host and port Default is RECEIVE. RECEIVE/TRANSMIT parameter is optional.
TCP	Indicates that TCP (Transmission Control Protocol) transport services should be used; TCP may also be specified as COTS
UDP	Indicates that *UDP* (*User Datagram Protocol*) transport services should be used; UDP may also be specified as CLTS The default is TCP. The transport protocol parameter is optional.
ASCII/EBCDIC	In transmit mode, this parameter specifies the character set to use when generating data to be sent. In receive mode, the ASCII/EBCDIC parameter is ignored. ASCII—ASCII character set EBCDIC—EBCDIC character set If not specified in transmit mode, the default is ASCII. The ASCII/EBCDIC parameter is optional

HOST(*host_name*)	In transmit mode, the HOST parameter specifies the name of the remote host to which TTCP transmits data. In receive mode, the HOST parameter is ignored.If not specified in transmit mode, the default is 127.0.0.1 (standard loopback Internet address). The HOST parameter is optional.
PORT(*port_number*)	Port number to use If not specified in transmit mode, the default is 9 (standard TCP/UDP discard port); if not specified in receive mode, the default is 2000. The PORT parameter is optional.
SYSID(*subsystem_ID*)	Subsystem ID of the TCPaccess address space other than the default used when TCPaccess was installed. SYSID may also be specified as SSN. The default subsystem ID is ACSS The SYSID parameter is optional.
BUFLEN(*buffer_length*)	Buffer length to be used In transmit mode, COUNT buffers of this length are transmitted. In receive mode, this is the length of each receive buffer. BUFLEN may also be specified as LENGTH If not specified, the default is 1024. The BUFLEN parameter is optional.
BUFNUM (*number_of_buffers*)	Number of buffers to receive into or transmit from simultaneously; that is, the queue depth of buffering If BUFNUM is greater than 1, there will be transmit or receive overlap, which is generally more efficient. BUFNUM may also be specified as NUMBER or BUFCT. The default is 5. The BUFNUM parameter is optional.
COUNT (*transmit_buffer_count*)	In transmit mode, COUNT specifies the number of buffers to transmit. COUNT may also be specified as CNT or CT. This parameter is silently ignored in receive mode. If not specified in transmit mode, the default is 1024.
TASKS (*number_of_receive_tasks*)	In receive mode, specifies the maximum number of receive subtasks that can be active to receive data from remote hosts. This equates to the maximum number of remote hosts that can be transmitting to the TTCP receive server simultaneously. TASKS may also be specified as SUBTASKS. This parameter is silently ignored in transmit mode. The TASK parameter is optional; if not specified in receive mode, the default is 6.

Examples

- The following is sample output of the TTCP program when in transmit mode:

```
TTCPT:     TRANSFER SECONDS   5.5
           TSEND'S           1024,       TSEND'S/SEC     185.5
           BYTES SENT        1048576,    BYTES/SEC       189959.4
```

- The following is sample output of the TTCP program when in receive mode:

```
TTCPR:    TRANSFER SECONDS   9.6
          TRECV'S           1024,       TRECV'S/SEC     34.6
          BYTES RECEIVED    048576,     BYTES/SEC       35448.8
```

WHOIS

The WHOIS TSO command processor acts as a WHOIS client and queries a default or user-specified WHOIS server to obtain information about a user-specified name. Information received from the WHOIS server is printed to your terminal. This command can be used to obtain information about hosts and individuals who use the network. Refer to RFC 954 for more information about the WHOIS service.

WHOIS NAME(*lookup_name*) [**HOST**(*server_host*)]
[**SYSID**(*subsystem_ID*)]

Syntax Description

WHOIS	Invokes the query to obtain information about the specified user.
NAME(*lookup_name*)	The name to use to obtain information. The *lookup_name* parameter can be a string of arbitrary characters to pass to the server to obtain information. Space characters should not be embedded in the *lookup_name* unless the whole parameter is enclosed in double quotes (e.g., NAME(YOGI BEAR@JELLYSTONE)). The NAME parameter is required.
HOST(*server_host*)	Allows the user to specify a WHOIS server host to use instead of the default. The *server_host* parameter should be the name of the host that provides the WHOIS service. The default host is NIC.DDN.MIL. The HOST parameter is optional.
SYSID(*subsystem_ID*)	Allows you to specify the subsystem ID of TCPaccess if the default was not used when TCPaccess was installed. The default subsystem ID is ACSS. The SYSID parameter is optional.

Example

The following is sample output of the WHOIS program:

```
Wells, David F.         (DW140)     dwells@LETTERKENN-EMH1.ARMY.MIL
CSDA-East
Attn: AMXLS-LILL
Letterkenny Army Depot
Chambersburg, PA 17201-4180(
(717) 267-8100 (DSN) 570-8100
Record last updated on 29-Sep-97
```

11

Programmable Batch Interface For FTP and FTP2

This chapter describes the programmable batch interface for FTP and FTP2. It includes these sections:

- "What Is the Programmable Batch Interface?" on page 11-2

  Describes how to use a programmable batch interface to FTP/FTP2.

- "FTPBATCH Interfaces" on page 11-3

  Describes the FTPBATCH module and the control fields.

- "Coding The Batch Application" on page 11-8

  Describes how to code the application.

- "Execution Samples" on page 11-10

  Describes batch application examples using FTP and FTP2.

- "Debugging Information" on page 11-12

  Describes information useful for debugging application programs.

- "Sample Batch Programs" on page 11-13

  Describes and lists several sample batch programs.

What Is the Programmable Batch Interface?

Using FTP/FTP2 in batch mode automates the process of transferring files, which is particularly useful if your site performs FTPs to/from particular servers repeatedly. Executing the FTP/FTP2 operations via a batch program can eliminate mistakes caused by typing errors and users only need to know how to start the batch program and don't need to learn or understand the FTP/FTP2 transfer commands.

For example, if you need to transfer a specific file from one host to another on a regular basis a batch program can be written to execute the commands automatically instead of having to repeat the same FTP/FTP2 commands from a terminal each time the file transfer is necessary.

Although both FTP and FTP2 can be invoked in batch mode existing batch methods cannot do much to respond to error conditions that may occur during a file transfer. A programmable batch interface has been added to TCPaccess that lets you write a program using a language of your choice (as long as the language supports an assembler language CALL interface, such as C, PL/I, COBOL, etc.) to issue FTP and FTP2 commands and to respond to conditions that may develop during the course of those FTP/FTP2 sessions.

- The batch program you write will run in a batch region on an MVS host.
- Your application will simulate input from a user's terminal.
- Multiple batch routines can be executed concurrently.
- You are limited only by the number of LUs defined to TCPaccess. In general, each FTP/FTP2 user ("user" includes a batch application) requires a minimum of two LUs for execution.

For details on invoking FTP and FTP2, and FTP3, read "Invoking Client FTP2" on page 2-5, "Invoking Client FTP3" on page 3-4, and "Invoking Client FTP" on page 4-5.

FTPBATCH Interfaces

The FTPBATCH module within TCPaccess is used as an interface between your batch program and TCPaccess, FTP, and FTP2 programs.

Note: TCPaccess has an alias of HB0007 for backwards compatibility. Batch programs using previous versions of this module will have to assemble/link the new module for successful execution.

When writing your batch program, you must set register R1 to point to the address of a parameter list. The first 4 bytes of the parameter list must point to an FTPBATCH batch control area and the last 4 bytes must point to an FTPBATCH data control area.

The batch control area is a 10-field control block that stores the TCPaccess commands you want executed. It also holds the responses that are returned from the TCPaccess application.

The data control area consists of a single 79-character field. This area is where your batch program must write the FTP/FTP2 commands you would ordinarily enter on a terminal while interactively operating FTP/FTP2 (for example, OPEN, GET, PUT, or APPEND).

Batch Control Fields

This section describes each field in the batch control area.

Command Field

The command field indicates the batch command that TCPaccess will execute. “Table 11-1” on page 11-3 shows the alphabetic characters that are valid values for this field:

Table 11-1 Valid Values for the Batch Command Field

Value	Description
O	**OPEN** command Establishes (opens) a connection between the batch application and an FTP or FTP2 application. Which application (FTP/FTP2) you establish a connection with is determined by what is specified on the SYSOPT input record in the startup JCL.
S	**SEND** command Indicates you are sending a command to the application (either FTP or FTP2). The actual command to be sent must be stored in the 79-character data control area.
R	**RECEIVE** command Receives a response from the FTP/FTP2 application. The response received will be the response from the previous command sent (via the **SEND** command). You are not required to issue a RECEIVE request for every message generated by the previous SEND. All response messages not retrieved via a RECEIVE will be ignored. (This allows the use of a batch program that only executes SEND requests.) All commands and responses will appear in the SYSPUT log file.

Value	Description
C	**CLOSE** command Closes the connection between your batch application and the FTP/FTP2 application.

"Table 11-2" on page 11-3 describes how to terminate FTP and FTP2:

Table 11-2 Terminating FTP and FTP2

Reason For Termination	FTP	FTP2
User program issues a **CLOSE**.	yes	yes
User program issues a **SEND** request that executes an **END** command.	yes	yes
User program issues a **SEND** request that executes a **BYE** or a **QUIT**.	no	yes

Although a CLOSE command is not always required after a BYE, END, or QUIT command, it is recommended that you issue a CLOSE request before exiting your program. If a user program CLOSE request is issued after a BYE, END, or QUIT, a return code of 16 (Close error - not open) will be returned to your program.

Length

The length field identifies the number of characters allowed as input from the data control area. Valid selections are 1 to 79. Your program must pad this field with spaces (X'40') before writing to it, to clean out remnants of previous commands.

Return Code

The return code field contains the return code received from TCPaccess indicating if a command was returned in error. “Table 11-3” on page 11-5 lists the possible return codes:

Table 11-3 Return Codes

Return Code	Description
0	Command completed successfully
3	Invalid command received on input
4	Invalid message length specified
5	Application no longer active
7	Invalid parameter input
8	Send command execution failed. Check SYSPUT output. See the note that follows this table
14	Logic error - next command must be open or close
15	Open error - already open
16	Close error - not open
17	Send error - not open
18	Receive error - not open

Note: When a user program issues a SEND request to execute an FTP command, there may be several response messages associated with a single SEND command. A return code of 08 will be returned to the user program in two ways:

- When control is returned to the user program following a user SEND request. This indicates the command did not complete successfully.
- When control is returned to the user program following a user RECEIVE request. Normally the 08 return code will be set on the message that FTP/FTP2 encountered the error. But in some cases (due to processing logic), the 08 return code will be returned to the user on a following message.

Control1

The control flag set by TCPaccess in this field indicates the relative position of the data block in a data stream. This flag is set to one of the values shown in “Table 11-4” on page 11-6:

Table 11-4 Control1 Flags

Flag	Description
F	Indicates the first block in the data stream
M	Indicates the middle block in the data stream
L	Indicates the last block in the data stream

Control2

This field contains a control flag set by TCPaccess that indicates the state of session activities. This flag will be set to one of the values shown in “Table 11-5” on page 11-6:

Table 11-5 Control2 Flags

Flag	Description
S	Indicates a session with TCPaccess is being started
R	Indicates the RESPONSE field contains a reply to a command (O/S/R/C)
Q	Indicates the session is closing (quit)

When FTP or FTP2 terminates with a BYE, END, or QUIT command, this field is set to Q when control is passed back to your program.

Secondary LU (Send)

This field is not used.

Secondary LU (Receive)

This field is not used.

Response

This field contains textual data returned from TCPaccess FTP or FTP2 application. This area must be defined as a 132-character area. It contains an image of a buffer sent to the SYSPUT output file upon returning to the user program after a RECEIVE request.

Data Control Field

When FTP/FTP2 commands are issued from a user's terminal, all data is on one line (the screen has only one line from which to enter single line commands). Therefore, when FTP/FTP2 commands are issued by a batch application, the data is restricted to one line containing up to 79 characters. The FTP/FTP2 commands you want issued must be written into the data control field.

Coding The Batch Application

Batch applications can be coded in any language capable of making assembly language calls. This includes languages such as COBOL, C, and PL/I. Your batch application must submit FTP/FTP2 commands according to the specifications described in “Batch Invocation” on page 2-6 in “Client FTP2”. Your batch application calls the TCPaccess FTPBATCH module, which passes the information to TCPaccess as if it came from the user’s terminal. Your application should also check the error return codes after issuing each batch command. (Read “FTPBATCH Interfaces” on page 11-3.)

After your batch application is written, it must be compiled, cataloged, and link-edited. The TCPaccess load library must be included with the link-edit job as module FTPBATCH is located in this library.

Your batch application (user program) issues calls to the FTPBATCH module which, in turn, gets input (i.e., the run time options) from the SYSOPT data set and passes that input to the FTP/FTP2 programs. Output from the FTP/FTP2 commands is directed to the SYSPUT file. TCPaccess controls the FTP/FTP2 programs, although this occurs transparently to your application.

Authorized Program Facility (APF)

If your batch program is to communicate with the FTP2 application, your program must reside in an authorized program library. To authorize a new LOAD file, you must add a new statement to the IEAAPFxx member in your SYS1.PARMLIB. After adding a new LOAD file to IEAAPFxx, the system must be IPLed to enable the new authorization. Also, your program must be link edited with the authorization parameter of SETCODE AC(1).

Note: Batch programs that work with the FTP application are not required to be APF authorized.

Application Selection (SYSOPT) and Invocation Options

User batch programs must select either the FTP or FTP2 application for processing commands. The selection is made by coding FTP or FTP2 beginning in column 1 of the SYSOPT input record (see the execution examples below).

The SYSOPT input record can be used to pass invocation options to the FTP or FTP2 application. Following the FTP or FTP2 selection characters, include a slash (/) followed by the options you want.

Examples

- Sample invocation with options:

FTP2 /TEST (this syntax passes the TEST option to FTP2)

- Sample invocation without options:

FTP2

You can also enter PL/I options along with FTP/FTP2 options. The PL/I options must be coded following the FTP or FTP2 selection characters, but before the separator (/).

Examples

The following are PL/I invocation examples:

FTP2 NOSPIE NOSTAE /TEST

(Note that the TEST option follows the PL/I options.)

FTP2 NOSPIE NOSTAE

Note: The FIOS option can not be used with the programmable batch interface.

User Program Batch Invocation

Your user program can run in batch like any other batch program, or as a TSO command processor by running under a batch Terminal Monitor Program (TMP). FTP2 commands DO, DIR, and LS, and the FTP DO command require a TSO environment to execute, thus they will only execute in batch under the TMP.

With batch, it is important to specify the FTP commands carefully, because the slightest error might cause all subsequent commands to fail and force you to rerun the batch job. Your program should test the return code after each request to ensure that correct processing is achieved.

Execution Samples

These examples are provided to help you code your applications.

Sample Using FTP

Your batch application can be invoked like any other batch program. This sample shows how to invoke your program to work with FTP.

```
//jobname      JOB   job-stmt-parms
//FTPSTEP      EXEC  PGM=user-prog
//STEPLIB      DD    DISP=SHR,DSN=user.linklib
//             DD    DSN=SNSTCP.Vxxx.LINK, /*IF FTP NOT LINKLISTED*/
//                   DISP=SHR
//SYSPUT       DD    SYSOUT=*,DCB=BLKSIZE=133
//SYSUDUMP     DD    SYSOUT=*
//SYSOPT       DD    *,DCB=BLKSIZE=80
FTP / option1 option2 ...
/*
```

Sample Using FTP2

You can set up your FTP2 application for either batch or TMP execution.

Batch Program

The following is an example of how to set up your FTP2 application for batch execution.

```
//jobname      JOB   job-stmt-parms
//FTPSTEP      EXEC  PGM=user-prog
//STEPLIB      DD    DISP=SHR,DSN=user.linklib
//             DD    DSN=SNSTCP.Vxxx.LINK, /*IF FTP2 NOT LINKLISTED*/
//                   DISP=SHR
//SYSPUT       DD    SYSOUT=*,DCB=BLKSIZE=133
//SYSUDUMP     DD    SYSOUT=*
//NETRC        DD    DISP=SHR,DSN=userid.FTP.NETRC
//SYSOPT       DD    *,DCB=BLKSIZE=80
FTP2 / option1 option2 ...
/*
```

Note: When you invoke FTP or FTP2 via a batch program, the TSO environment required for the **DO**, **DIR**, and **LS** commands is not available. Consequently, these commands will be rejected when FTP or FTP2 executes via a batch program.

The **NETRC** statement is required if the automatic login feature is being used.

TMP Execution

Your user program can be invoked in batch as a TSO command processor by running it under a batch TMP. This mode of operation allows your program to use **DO**, **DIR**, and **LS** commands (which are not supported in ordinary batch mode). TMP execution also causes output from TSO commands to be directed to the **SYSTSPRT** file; they are not returned to your batch program.

The following example shows how to initiate your batch program under TMP:

```
//jobname     JOB   job-statement-parameters
//TPMSTEP     EXEC  PGM=IKJEFT01
//STEPLIB     DD    DISP=SHR,DSN=user.load.lib
//            DD    DSN=SNSTCP.Vxxx.LINK, /*IF NOT LINKLISTED*/
//                  DISP=SHR
//SYSPRINT    DD    SYSOUT=*
//SYSPUT      DD    SYSOUT=*,DCB=BLKSIZE=133
//SYSTSPRT    DD    SYSOUT=*
//SYSUDUMP    DD    SYSOUT=*
//SYSTSIN     DD    *
user_program_name
/*
//SYSOPT      DD *
[FTP | FTP2] / option1 option2 ...
/*
```

Debugging Information

This section describes information to help you debug your application programs.

Environment

The environment used to execute the programmable batch facility consists of these three parts:

- Your batch application
- The FTPBATCH interface module
- The TCPaccess FTP or FTP2 application

These three parts all reside within the same MVS address space during execution. Therefore, be aware that program errors within your application can overlay code within the FTPBATCH or FTP/FTP2 application causing unpredictable results. Ensure that your applications do not conflict with the FTPBATCH or FTP/FTP2 application processing.

PL/I Environment

The FTP and FTP2 applications are coded in the PL/I language. The environment setup when PL/I is used can cause some problems when trying to debug other modules of a different language that may be running within the same address space. To help debugging, use PL/I execution parameters to control PL/I debug options. PL/I options can be submitted by entering them on the SYSOPT input record.

Example

FTP NOSPIE NOSTAE / TEST

FTP and FTP2 Debugging Options

Both the FTP and FTP2 applications have debugging options that let you obtain debugging information on the internal operation of the user FTP or FTP2 program and the interactions between the User FTP program and the Server FTPs. When using debugging options, trace information generated by the option will be sent to both the SYSPUT file and to your batch program.

Message Output

Output messages created from the programmable batch facility can be found at several separate locations within the jobs output stream. The SYSGET output file will contain FTP or FTP2 transfer and trace related information. WTO messages are on the MVS console log and can be directed to the user's JCL message log file. Depending on the language used to write your batch application, the messages output by the language being used may appear in a separate area of your output. In general, be aware that the complete job output must be reviewed to assist you if problems occur.

Sample Batch Programs

This section contains sample batch applications to illustrate the requirements for coding batch applications. The samples provided are for reference only and may not contain sufficient details for an actual TCPaccess environment.

COBOL Language Batch Application

See member FTPCOBOL in the SAMP library for a sample of a COBOL language batch application.

Assembler Language Batch Application

See member FTPASM in the SAMP library for a sample of an Assembler language batch application.

Sample Batch Application Linkedit

```
//LINK1   DD  DISP=SHR,DSN=SNSTCP.Vxxxx.LINK
//LINK2   DD  DISP=SHR,DSN=user.linklib
//SYSLIN DD *
   ENTRY user_prog
   INCLUDE LINK1(FTPBATCH)
   INCLUDE LINK2(user_prog)
   MODE AMODE(31),RMODE(24)
   NAME  user_prog(R)
/*
```

Sample Batch Application SYSPUT Output

```
14:05:00 TCPaccess User FTP - Enter command or '?'
14:05:00 USERFTP: a:conn 123.456.7.8
14:05:09 A: UNIX.COMPANY.COM -- FTP Server, Enter command or HELP
14:05:09 USERFTP: a:log wv85 #######
14:05:11 USERFTP: b:conn 234.567.8.9
14:05:13 B:220 unix FTP server (SunOS 4.1) ready.
14:05:13 USERFTP: b:log unix #######
14:05:14 USERFTP: a:get filea 'TEST.LIB(MEMBER)'
14:05:14 HOST PREFIX IGNORED FOR GET
14:05:16 B:150 ASCII data connection for filea (234.567.8.9,4175) (428 bytes).
14:05:16 B:226 ASCII Transfer complete.
14:06:08 A:150-Dataset open with attributes:
14:06:08 A:Type A N    Tabs    8   Stru F   Mode S   Recall 5
14:06:08 A:Path TEST.LIB(MEMBER)
14:06:08 A:Volser REST00   Unit SYSDA   Dsorg   PO   Recfm FB   Lrecl   80
14:06:08 A:Blksize   3120   Space 10 0 Tracks   Rlse   Dir   5
14:06:08 A:150
```

```
14:06:08 A:226-Transfer complete
14:06:08 A:439 bytes received in 0.85 seconds (516 bytes/s)
14:06:08 A:Path TEST.LIB(MEMBER) User TEST Data bytes received 417
14:06:08 A:Disk tracks written 1   Records padded    11
14:06:08 A:226
14:06:09 USERFTP: a:quit
14:06:10 A:221 Session terminated
14:06:11 USERFTP: b:quit
14:06:11 B:221 Goodbye.
14:06:12 USERFTP: END
```

Index

Symbols

! command, Client FTP2 2-18
$ command, Client FTP2 2-19
? command
  Client FTP 4-16
  Client FTP2 2-17
  Client FTP3 3-24

A

A=B command, Client FTP 4-51
A?B command, Client FTP 4-52
ABOR command
  Client FTP 4-17
  Client FTP2 2-20
ACCOUNT command, Client FTP3 3-25
ACK 8-4
ACSHELLO program 10-3
ADD command, Client FTP 4-18
addressing mail 7-4
ALLO command
  Client FTP 4-19
  Client FTP2 2-21
  Server FTP 5-9
allocating
  disk space 5-32
  disk space, fragmentation 5-32
  extents 5-32
  files 4-19
  secondary disk space 5-24
  space in tracks 5-24
API and SOCKET applications
  ACSHELLO 10-3
  FINGER TSO command processor 10-4
  TTCP TSO command processor 10-5
  WHOIS TSO command processor 10-8
APP invocation option
  Client FTP 4-9
  Client FTP2 2-8
APPE
  append operation rules 5-36
  data transfer option 5-39
APPEND command
  Client FTP2 2-22
  Client FTP3 3-26
appending files 4-18
APPLID, in TSO TELNET 6-4
ASA format 5-44, 5-47
  carriage control 5-41
  retrieving line image files 5-45
ASCII
  format effectors 5-41
  terminal types supported for TELNET 6-16
ASCII command
  Client FTP2 2-23
  Client FTP3 3-27
Assembler language batch applications 11-13
ATTR parameter, SITE command 5-17
AUTOINDEX parameter, SITE command 5-17
Autologon, for VTAM applications 6-19
automatic logon, See Client FTP2
AUTOMOUNT parameter of FTP.DATA 3-14
AUTOMOUNT parameter, SITE command 5-17
AUTORECALL parameter of FTP.DATA 3-14
AUTORECALL parameter, SITE command 5-17

B

backslash (\), for continuing long lines 9-7
batch applications in Assembler 11-13
BINARY command
  Client FTP2 2-24
  Client FTP3 3-28
binary file support for Server FTP 5-2
BLOCK invocation option, Client FTP3 3-4
block mode 2-48, 3-49, 4-33
  in the MODE command 3-49
BLOCK parameter, SITE command 5-17
block size 5-17
BLOCKSIZE parameter of FTP.DATA 3-14
break signal, in TELNET 6-10
BYE command
  Client FTP 4-20
  Client FTP2 2-25

C

cancelling a command, See ABOR command
cancelling SITE SUBMIT 5-24
CARDS parameter, SITE command 5-17, 5-20
catalog 5-32
  using with FTP 5-5
cataloging data sets 5-5
CCONNTIME parameter of FTP.DATA 3-14
CD command 2-26
  Client FTP3 3-29
CD parameter, SITE command 5-17
CDUP command
  Client FTP 4-21
  Client FTP2 2-27
  Client FTP3 3-30
central file directory 5-32
change directory command 4-23
character translation mode 5-25
character type rules 5-42
  file structure 5-42
CHARSET parameter, SITE command 5-17
CHKPTint parameter of FTP.DATA 3-14
CHKPTINT parameter, SITE command 5-17
Client FTP
  command conventions 4-12
  commands 4-14
    ? command 4-16
    A=B 4-51
    A?B 4-52
    ABOR 4-17
    ADD 4-18
    ALLO 4-19
    BYE 4-20
    CDUP 4-21
    CONN 4-22
    CWD 4-23
    DELE 4-24
    DO 4-25
    END 4-26
    EXPE 4-27
    FTP TSO 4-5
    GET 4-28
    HELP 4-29
    LIST 4-30
    LOG 4-31
    MKD 4-32
    MODE 4-33
    NLST 4-35
    PUT 4-36
    PWD 4-37
    QUIT 4-38
    QUOT 4-39
    REN 4-40
    REST 4-41
    RMD 4-43
    SEND 4-44
    SITE 4-45
    SNDS 4-46
    STAT 4-47
    STRU 4-48
    TSO CALL 4-5
  debugging options 4-10
    DISP 4-10
    TEST 4-11
    TESTI 4-11
    VLT 4-11
  definition of 4-3
  file transfer examples 4-54
  handling TELNET connections 4-10

host prefixes 4-13
invocation options 4-9
APP 4-9
FIOS 4-9
LOGT 4-10
SYS 4-10
WAIT 4-10
invoking
as a batch program 4-7
through TSO 4-5
under batch TMP 4-8
path names 4-12
program condition code 4-7
transmission modes 4-33
compressed mode 4-33
stream mode 4-33
using 4-12
Client FTP2 2-22
APP invocation option 2-8
batch invocation 2-6
batch mode, program condition code 2-6
command conventions 2-11
command summary 2-15
commands
! 2-18
$ 2-19
? 2-17
ABOR 2-20
ALLO 2-21
ASCII 2-23
BINARY 2-24
BYE 2-25
CD 2-26
CDUP 2-27
CLOSE 2-28
DEBUG 2-29
DELETE 2-30
DIR 2-31
DISCONNECT 2-32
DO 2-33
EBCDIC 2-34
END 2-35
EXPE 2-36
GET 2-39
HELP 2-40
LOG 2-41
LQUOTE 2-42
LS 2-43
MACDEF 2-45
MKDIR 2-47
MODE 2-48
NTRANS 2-49
OPEN 2-50
PUT 2-51
PWD 2-52
QUIT 2-53
QUOTE 2-54
RECV 2-55
REMHELP 2-56
REMSITE 2-57
REMSNDS 2-58
REMSTAT 2-59
RENAME 2-60
REST 2-61
RMDIR 2-63
SEND 2-64
SITE 2-65
SNDS 2-66
STATUS 2-67
STRUCT 2-68
SUNIQUE 2-69
TYPE 2-70
debugging options 2-10
file transfer examples 2-73
FTP2 restart marker 2-76
FIOS invocation option 2-8
high-level qualifier 2-14
HOST invocation option 2-9
invoking 2-5
as a batch program 2-7
LOGT invocation option 2-9
MODE command
block mode 2-48
stream mode 2-48
NETRC file 2-13
specifying 2-6
NOA invocation option
overview 2-3
path names 2-11
SYS invocation option 2-9
TEST1 debugging option
testing control connections 2-12
TSO CALL command 2-6
using 2-11

VLT debugging option 2-10
Client FTP3
batch invocation 3-7
command conventions 3-18
commands 3-22
? 3-24
ACCOUNT 3-25
APPEND 3-26
ASCII 3-27
BINARY 3-28
CD 3-29
CDUP 3-30
CLOSE 3-31
DEBUG 3-32
DELETE 3-33
DELIMIT 3-34
DIR 3-35
EBCDIC 3-36
GET 3-37
HELP 3-38
LCD 3-39
LMKDIR 3-40
LOCSITE 3-41
LOCSTAT 3-42
LPWD 3-43
LS 3-44
MDELETE 3-45
MGET 3-46
MKDIR 3-48
MODE 3-49
MPUT 3-51
NOOP 3-52
OPEN 3-53
PASS 3-54
PUT 3-55
PWD 3-56
QUIT 3-57
QUOTE 3-58
RENAME 3-59
RESTART 3-60
RMDIR 3-61
SENDSITE 3-62
SITE 3-63
STATUS 3-64
STRUCT 3-65
SUNIQUE 3-65
SYSTEM 3-66
TRACE 3-67
TSO 3-68
TYPE 3-69
CPU Utilization 3-2
data transfer 3-2
illustrated 3-3
introduction 3-2
invocation options
BLOCK 3-4
EXIT 3-4
LOCALHOST 3-5
SSID 3-5
TCP 3-5
TRACE 3-5
TRANSLATE 3-5
VERBOSE 3-5
invocation optionsDEBUGDEBUG invocation option, Client FTP3 3-4
invoking 3-4
through batch program 3-7
MODE command
block mode 3-49
stream mode 3-49
path names 3-17
TSO CALL command 3-6
TSO invocation 3-4
two-party model illustrated 3-3
using 3-17
Client TELNET
3767 terminal 6-3
allocating SYSPRINT file 6-4
break signal 6-10
character-oriented mode 6-3
command entry rules 6-8
commands
APPLID string 6-10
ATTN 6-12
AUTO ON|OFF 6-12
BRK 6-10
CLEAR 6-13
controlling input and output 6-10
CURSOR 6-8, 6-13
ECHO ON|OFF 6-11
END|BYE 6-10
EXEC 6-13
for sending data 6-9
for session control 6-10

HELP 6-13
HEX 6-10
HIDE 6-12
IP 6-10
KO 6-9
LOG userid 6-13
NOTE 6-12
PRINT 6-11
READ dsname 6-11
RESHOW 6-13
RETURN 6-10
RTO seconds 6-11
SAMPLE 6-11
SEND 6-7, 6-9
SLEEP 6-12
TEST ON|OFF 6-13
TSO|DO 6-13
TTO number 6-10
WAIT milliseconds 6-11
WRITE dsname 6-11
XCTL 6-9
XESC 6-9
XNNL 6-9
XWNL 6-9
debugging options, U 6-5
function keys 6-8
half-duplex locked-keyboard operation 6-3
invoking
from an ASCII terminal 6-17
ISPF with 6-3
line mode operation 6-5
line-by-line operation 6-3
methods of using 6-3
NULL transaction 6-7
NVT operation 6-3
overview of 6-2
retransmitting data 6-8
screen mode operation 6-6
sending data 6-7
sending data lines 6-5
split-screen operation 6-3
TELNET command
APPLID 6-4
LINE 6-4
options 6-4
SYS 6-4
TTY 6-4
TEST debugging option 6-5
Client TELNET for ASCII terminals 6-16
CLOSE command
Client FTP2 2-28
Client FTP3 3-31
COBOL batch applications 11-13
coding batch applications 11-8
batch invocation 11-9
command conventions
Client FTP 4-12
Client FTP2 2-11
Client FTP3 3-18
command entry rules, Client TELNET 6-8
commands
ALLO, Server FTP 5-9
APPEND 2-22
Client FTP 4-14
? command 4-16
A=B 4-51
A?B 4-52
ABOR 4-17
ADD 4-18
ALLO 4-19
BYE 4-20
CDUP 4-21
CONN 4-22
CWD 4-23
DELE 4-24
DO 4-25
END 4-26
EXPE 4-27
GET 4-28
HELP 4-29
LIST 4-30
LOG 4-31
MKD 4-32
MODE 4-33
NLST 4-35
PUT 4-36
PWD 4-37
QUIT 4-38
QUOT 4-39
REN 4-40
REST 4-41
RMD 4-43
SEND 4-44

SITE 4-45
STAT 4-47
STRU 4-48
TSO 4-5
TSO CALL 4-5
Client FTP2
! 2-18
$ 2-19
? 2-17
ABOR 2-20
ALLO 2-21
APPEND 2-22
ASCII 2-23
BINARY 2-24
BYE 2-25
CD 2-26
CDUP 2-27
DEBUG 2-29
DELETE 2-30
DIR 2-31
DISCONNECT 2-32
DO 2-33
EBCDIC 2-34
END 2-35
EXPE 2-36
GET 2-39
HELP 2-40
LOG 2-41
LQUOTE 2-42
LS 2-43
MACDEF 2-45
MKDIR 2-47
MODE 2-48
NTRANS 2-49
OPEN 2-50
PUT 2-51
PWD 2-52
QUIT 2-53
QUOTE 2-54
RECV 2-55
REMHELP 2-56
REMSITE 2-57
REMSNDS 2-58
REMSTAT 2-59
RENAME 2-60
REST 2-61
RMDIR 2-63
SEND 2-64
SITE 2-65
SNDS 2-66
STATUS 2-67
STRUCT 2-68
SUNIQUE 2-69
TYPE 2-70
Client FTP3 3-22
? 3-24
ACCOUNT 3-25
APPEND 3-26
ASCII 3-27
BINARY 3-28
CD 3-29
CDUP 3-30
CLOSE 3-31
DEBUG 3-32
DELETE 3-33
DELIMIT 3-34
DIR 3-35
EBCDIC 3-36
GET 3-37
HELP 3-38
LCD 3-39
LMKDIR 3-40
LOCSITE 3-41
LOCSTAT 3-42
LPWD 3-43
LS 3-44
MDELETE 3-45
MGET 3-46
MKDIR 3-48
MODE 3-49
MPUT 3-51
NOOP 3-52
OPEN 3-53
PASS 3-54
PUT 3-55
PWD 3-56
QUIT 3-57
QUOTE 3-58
RENAME 3-59
RESTART 3-60
RMDIR 3-61
SENDSITE 3-62
SITE 3-63
STATUS 3-64

STRUCT 3-65
SUNIQUE 3-65
SYSTEM 3-66
TRACE 3-67
TSO 3-68
TYPE 3-69
FTP3 TSO command 3-4
Server FTP
HELP 5-10
MKD 5-12
REST 5-13
RMD 5-14
SITE 5-15
STAT 5-27
COMPACT parameter, SITE command 5-17
compressed mode 3-50, 4-33
CONDDISP parameter, SITE command 5-17
CONN command, Client FTP 4-22
connections, show connections command 4-52
continuing lines with backslash (\) 9-7
control characters
HT 5-40
VTAM TELNET 6-15
control connections, testing in Client FTP2 2-12
control file format for USPOOL 8-6
controlling input and output, in TELNET 6-10
CPU utilization 3-2
CPU utilization with FTP3 3-2
creating new directories 4-32
in Server FTP 5-12
CURSOR command, in TELNET 6-8
CWD command, Client FTP 4-23
CYLINDER parameter, SITE command 5-17

D

DACLASS parameter, SITE command 5-18
data control blocks 5-33
data file format for USPOOL 8-6
data sets
append operations 5-36
attribute errors 5-36
attribute format rules 5-36
attributes 5-28
default 5-34
data set names 5-28
Generation Data Group (GDG) 5-29
multi-volume 5-33
naming conventions 5-28
partitioned 5-31, 5-34
sequential 5-34
tape on remote hosts 5-6
units and volumes 5-28
data transfer
acknowledgment (ACK) 8-4
APPE 5-39
in Client FTP2 2-12
operations 5-39
restarting 5-13
RETR 5-39
STOR 5-39
DATACLASS parameter of FTP.DATA 3-14
DATACLASS parameter, SITE command 5-18
DATACTTIME parameter of FTP.DATA 3-14
datagram, IP header 1-3
DATASETMODE parameter, SITE command 5-18
DATASETPREFIX parameter of TCPIP.DATA 3-13
DBCSSET parameter, SITE command 5-18
DCBDSN parameter of FTP.DATA 3-14
DCBDSN parameter, SITE command 5-18
DCLOSE parameter, SITE command 5-18
DCONNTIME parameter of FTP.DATA 3-14
DEBUG command
Client FTP2 2-29
Client FTP3 3-32
debugging
batch applications 11-12
Client FTP 4-10
Client FTP2 2-10
VLT option 2-10
options in TELNET 6-5
U option in TELNET 6-5
DELE command, Client FTP 4-24
DELETE command
Client FTP2 2-30
Client FTP3 3-33
deleting a directory, Server FTP 5-14

deleting files 4-24
DELIMIT command, Client FTP3 3-34
DEVNULL parameter, SITE command 5-18
DIDLE parameter, SITE command 5-18
DIR command
  Client FTP2 2-31
  Client FTP3 3-35
DIR parameter, SITE command 5-18
directory
  change directory command 4-23
  make directory command 4-32
  remove directory command 4-43
DIRECTORY parameter of FTP.DATA 3-14
DIRECTORYMODE parameter of FTP.DATA 3-15
DIRECTORYMODE parameter, SITE command 5-19
DISCONNECT command, Client FTP2 2-32
disconnecting from a host 4-38
disk format (DCB) attributes 5-33
disk space
  allocating 5-32
  allocating in Server FTP 5-24
displaying current working directory 4-37
DO command
  Client FTP 4-25
  Client FTP2 2-33
domain literal 7-4, 7-12
Domain Name Resolver 7-4
Domain Name System 1-5
DOMAINORIGIN parameter of TCPIP.DATA 3-13
DOPEN parameter, SITE command 5-19
DSEQ parameter, SITE command 5-19
DUMMY parameter, SITE command 5-19

E

EBCDIC command
  Client FTP2 2-34
  Client FTP3 3-36
END command
  Client FTP 4-26
  Client FTP2 2-35
envelope 7-4
escape sequences, for TELNET 6-20
EXIT invocation option, Client FTP3 3-4
EXPDT, SITE command 5-19
EXPE command
  Client FTP 4-27
  Client FTP2 2-36
experimental commands 4-27
extents, space allocation 5-32

F

file handling
  Server FTP 5-3
    overriding default parameters 5-3
    transferring to a host 5-3
  Server FTP, records too long 5-3
  sophisticated file handling with Server FTP 5-4
file structure
  ASA format 5-44
    line image files 5-45
    retrieving raveled files 5-45
  line image files 5-42
  no format control 5-42
  TELNET format effectors 5-44
file system support for Server FTP 5-2
file transfer
  Client FTP examples 4-54
  in Client FTP 4-54
File Transfer Protocol (FTP) 4-3
  definition of 2-3
file transmission, restarting 2-72, 4-53
files
  appending 4-18
  line image 5-40
  NETRC 2-13, 3-19
  raveled 5-39
  record structured 5-40
  with fixed-length records 5-40
FILETYPE parameter of FTP.DATA 3-15
FILETYPE parameter, SITE command 5-20
FINGER TSO command processor 10-4
FIOS invocation option, Client FTP 4-9
FIREWALL option for Client FTP2 2-37
firewall option for FTP2 2-9

FMID LPIX1.50 4-7
folding/truncating records 5-39
formatting
  disks 5-33
  line feeds 8-9
FORTRAN parameter, SITE command 5-17, 5-20
fragmented disk space 5-32
FTP
  definition of 2-3
  magnetic tape support
    catalog usage 5-5
    cataloging data sets 5-5
    description of 5-4
    FTP Statement 5-4
    LABEL parameter 5-4
    MOUNT parameter 5-4
    mount request 5-6
    preventing timeouts 5-6
    SITE command parameters 5-5
    tape data sets on remote hosts 5-6
    writing multiple data sets to tape 5-5
  programmable batch interface 11-1
FTP server
  connections to 3-2
FTP TSO command, Client FTP 4-5
FTP.DATA 3-9, 3-11, 3-14
FTP2
  FIREWALL option 2-37
  NOFIRE option 2-9
  programmable batch interface 11-1
  restart marker example 2-76
FTP3 TSO command 3-4
FTPBATCH batch control fields 11-3
full-duplex connections 1-4
FULLTRK parameter SITE command 5-20
function keys, in TELNET 6-8

G

GAT TYPE(TAPE) 5-24
GAT, See Generic Attributes Table
gateways, packet-switching hosts 1-3
generation data sets 5-29
Generic Attributes Table (GAT) 5-4
generic units
  for creating new data sets 5-25
  specifying for output data set 5-25
GET command
  Client FTP 4-28
  Client FTP2 2-39
  Client FTP3 3-37

H

HALFTRK parameter, SITE command 5-20
HELP command
  Client FTP 4-29
  Client FTP2 2-40
  Client FTP3 3-38
  Client TELNET 6-13
  Server FTP 5-10
  Server TELNET 6-18
HFS parameter, SITE command 5-20
high-level qualifier, for Client FTP2 2-14
Host invocation option for FTP2 2-9
host name strings
  format of 1-5
  fully qualified or unqualified 1-5
  host parameter 1-5
  Internet host numbers 1-5
  port option 1-6
host names 6-14
  in mail 7-4
  mapping 7-11
host prefixes
  Client FTP 4-13
  switching 4-51
HOSTNAME parameter of TCPIP.DATA 3-13
HT control character 5-40

I

IBM Configuration Data Sets 3-9
IBUF parameter, SITE command 5-20
ICMP, definition 1-3
INACTTIME parameter of FTP.DATA 3-15
INIT macro 2-13
initiating batch applications through TMP 11-11
Internet-specific protocols, See protocols
interpreting hexadecimal field characters 6-10
invoking
  batch applications 11-9
  Client FTP 4-5, 4-8, 4-9
    as a batch program 4-7
  Client FTP2 2-5, 2-10
    APP option 2-8
    as a batch program 2-6
    FIOS option 2-8
    HOST option 2-9
    LOGT option 2-9
    NOA option
    SYS option 2-9
  Client FTP3 3-4
    as a batch program 3-7
  Client TELNET from an ASCII terminal 6-17
  SNDMSG 7-6
  TSO CALL command 2-6, 3-6
IP
  description of 1-3
  fragmentation 1-3
  host addressing 1-3
  packet reassembly 1-3
IRBLKSIZE parameter SITE command 5-20
IRLRECL parameter, SITE command 5-20
IRRECFM parameter, SITE command 5-20
ISPF, using with Client TELNET 6-3
ISPFENQ parameter, SITE command 5-20
ISPFRES parameter, SITE command 5-20

J

JES internal reader support for Server FTP 5-2, 5-37
JESLRECL parameter, SITE command 5-20
JESRECFM parameter, SITE command 5-20

L

LABEL parameter, SITE command 5-20
LCD command, Client FTP3 3-39
libraries
  PL/I runtime 3-2
line image files 5-40, 5-45
  storing 5-42
line logical record length 5-17
line mode operation, in TSO Client TELNET 6-5
LINE parameter, SITE command 5-17, 5-21
line printer daemon 8-3
  port 515 8-3
line-by-line operation, in Client TELNET 6-3
Linkedit 11-13
LIST command, Client FTP 4-30
listening ports, port 515 8-3
LISTFMT parameter, SITE command 5-21
LKEDRES parameter, SITE command 5-21
LMKDIR command, Client FTP3 3-40
LOADLIB parameter, SITE command 5-17, 5-20
LOCALHOST invocation option, Client FTP3 3-5
LOCSITE command, Client FTP3 3-41
LOCSTAT command, Client FTP3 3-42
LOG command
  Client FTP 4-31
  Client FTP2 2-41
logical record length 5-17, 5-33
LOGT invocation option
  Client FTP 4-10
  Client FTP2 2-9
lpr commands 8-3
lpr protocol 8-4
  hexadecimal values 8-4
LPWD command, Client FTP3 3-43

LQUOTE command, Client FTP2 2-42
LRECL parameter of FTP.DATA 3-15
LRECL parameter, SITE command 5-21
LS command
  Client FTP2 2-43
  Client FTP3 3-44

M

MACDEF command, Client FTP2 2-45
macros, INIT 2-13
mail
  addresses 7-4
  definition of terms 7-4
  domain literal 7-4
  Domain Name Resolver 7-4
  envelope 7-4
  host names 7-4
  mailbox name 7-5
  message header 7-4
  multi-homed host 7-4
  overview 7-2
  receiving 7-5, 7-8
  sending 7-6, 7-9
  signature 7-5
  SSMPT 7-8
  transport 7-2
mail transport
  domain literal 7-4
  Domain Name Resolver 7-4
  envelope 7-4
  host names 7-4
  message header 7-4
  multi-homed host 7-4
  post office 7-2
  SNDMSG 7-2
  SPOOL#4 7-2, 7-9
  SSMPT 7-8
  SSMTP 7-2
  USMTP 7-2, 7-11
mailbox name, definition of 7-5
make directory command 4-32, 5-12
MANAGEMENTCLASS parameter, SITE command 5-21
mapping host names 7-11
MDELETE command, Client FTP3 3-45
message header 7-4
MESSAGECASE parameter of TCPIP.DATA 3-13
MGET command, Client FTP3 3-46
MGMTCLASS parameter of FTP.DATA 3-15
MIGRATEVOL parameter of FTP.DATA 3-15
MIGRATEVOL parameter, SITE command 5-21
MKD command
  Client FTP 4-32
  Server FTP 5-12
MKDIR command
  Client FTP2 2-47
  Client FTP3 3-48
MODE command
  Client FTP 4-33
  Client FTP2 2-48
  Client FTP3 3-49
  stream mode 2-48, 3-49
MOUNT parameter, SITE command 5-21
mount request and file transfer to tape 5-6
MPUT command, Client FTP3 3-51
multi-homed host 7-4
  logic of 7-11
multi-volume data sets 5-25, 5-33
MVS
  catalog 5-32
  central file directory 5-32
  disk space allocation 5-32
MVS parameter, SITE command 5-20
MX record 7-11
MYOPENTIME parameter of FTP.DATA 3-15

N

NCP parameter of FTP.DATA 3-15
NCP parameter, SITE command 5-21
NDAB parameter, SITE command 5-21
NETRC file 2-6, 2-13, 3-19
  with REMCMND 9-4
NETRC invocation option for Client FTP2, See Client FTP2
NETSTAT command 6-18

Network Virtual Terminal (NVT), in TELNET 6-3
NLST command, Client FTP 4-35
NLSTCASE parameter, SITE command 5-22
NOA invocation option
NOFIRE option for Client FTP2 2-9
NOOP command, Client FTP3 3-52
NTRANS command, Client FTP2 2-49
null line command, in TELNET 6-9
NULL transaction, in Client TELNET 6-7

O

OBJECT parameter, SITE command 5-17, 5-20
OBUF parameter, SITE command 5-22
octets, as measurement 1-4
OPEN command
  Client FTP2 2-50
  Client FTP3 3-53
overriding default parameters, in Server FTP 5-3
OVERWRITE parameter, SITE command 5-22

P

packet-switching hosts 1-3
PAD parameter, SITE command 5-22
padding
  records 5-40
  stripping from fixed-length logical records 5-24
PARALLELMOUNT parameter, SITE command 5-22
partitioned data sets (PDS) 5-31, 5-34
PASS command, Client FTP3 3-54
path names 2-11
  in Client FTP 4-12
  specifying 4-30
path names, in Client FTP3 3-17
PDSE parameter, SITE command 5-22
PERSIST parameter, SITE command 5-22
PL/I FMIDs 4-7
PL/I runtime libraries 3-2
PL/I Transient (runtime) Library 6-3
POP parameter, SITE command 5-23
port option for host name strings 1-6
PORT parameter of FTP.DATA 3-15
post office, definition of 7-2
preventing timeouts, in file transfer to tape 5-6
PRIMARY parameter of FTP.DATA 3-15
PRIMARY parameter, SITE command 5-22
print commands for UNIX 8-3
PRINT parameter, SITE command 5-17, 5-20, 5-22
printing
  carriage return 8-9
  line feeds 8-9
  UNIX 8-3
PRIVATE parameter, SITE command 5-22
problem resolution in batch applications 11-12
program condition code 4-7
programmable batch applications
  application selection (SYSOPT) 11-9
  coding 11-8
  debugging 11-12
  execution samples 11-10
  FTPBATCH program 11-3
  invocation options 11-9
  Linkedit 11-13
  overview 11-2
  problem resolution 11-12
  sample programs 11-13
  SYSPUT output 11-13
  TMP 11-11
programmable batch interface for FTP and FTP2 11-1
programmable function keys 6-8
protocols 1-3
  ACK 8-4
  ICMP 1-3
  illustration 1-3
  IP 1-3
  IP, defined 1-3
  provided 1-3
  UDP 1-4
PUSH parameter, SITE command 5-23
PUT command
  Client FTP 4-36
  Client FTP2 2-51

Client FTP3 3-55
PWD command
Client FTP 4-37
Client FTP2 2-52
Client FTP3 3-56

Q

QDISK parameter, SITE command 5-23
QUIT command
Client FTP 4-38
Client FTP2 2-53
Client FTP3 3-57
QUOT command, Client FTP 4-39
QUOTE command
Client FTP2 2-54
Client FTP3 3-58

R

raveled files 5-39, 5-43
retrieving 5-45
storing 5-45
RDW parameter of FTP.DATA 3-15
RDW parameter, SITE command 5-23
RECALL parameter, SITE command 5-23
receiving mail 7-5, 7-8
RECFM parameter of FTP.DATA 3-15
RECFM parameter, SITE command 5-17, 5-23
record format 5-17, 5-33
data set attribute rules 5-36
padding fixed-length 5-40
record structure
ASA format 5-47
STRU R command 5-47
support for Server FTP 5-2
record structured file 5-40, 5-46
RECV command, Client FTP2 2-55
regular directory commands 4-27
REMCMND program 9-2
REMHELP command, Client FTP2 2-56
Remote Executor
REMCMND 9-2
REXEC daemon 9-2
TCPREXEC 9-2
using an NETRC file 9-4
remote printer definition 8-3
remove directory command 4-43
removing a directory 5-14
REMSITE command, Client FTP2 2-57
REMSNDS command, Client FTP2 2-58
REMSTAT command, Client FTP2 2-59
REN command, Client FTP 4-40
RENAME command
Client FTP2 2-60
Client FTP3 3-59
REPLYFMT parameter, SITE command 5-23
RES parameter, SITE command 5-23
resend site parameters command 4-46
REST command
Client FTP 4-41
Client FTP2 2-61
Server FTP 5-13
RESTART command, Client FTP3 3-60
restart markers 4-41, 4-53
RESTART parameter, SITE command 5-24
restarting a data transfer 5-13
restarting a file transfer 2-72, 4-41, 4-53
RETPD parameter of FTP.DATA 3-15
RETPD parameter, SITE command 5-19, 5-24
RETR, data transfer option 5-39
retransmitting data, in TELNET 6-8
retrieving
files 5-39
line image files 5-45
logical records 5-47
raveled files 5-45
REXEC
daemon 9-2
running in batch mode 9-6
RLSE parameter, SITE command 5-24
RMD command
Client FTP 4-43
Server FTP 5-14
RMDIR command
Client FTP2 2-63
Client FTP3 3-61

S

screen mode operation in TELNET 6-3, 6-6
SECONDARY parameter of FTP.DATA 3-15
SECONDARY parameter, SITE command 5-24
segments 1-4
SEND command 6-7
  Client FTP 4-44
  Client FTP2 2-64
sending data in Client TELNET 6-9
sending data lines in line mode 6-5
sending mail 7-6, 7-9
SENDSITE command, Client FTP3 3-62
sequential data sets 5-34
Server FTP
  binary file support 5-2
  commands
    ALLO 5-9
    HELP 5-10
    MKD 5-12
    REST 5-13
    RMD 5-14
    SITE 5-15
    STAT 5-27
  file handling 5-3
    sophisticated 5-4
    transferring to a host 5-3
  file system support 5-2
  JES internal reader support 5-2
  overview of 5-2
  path name syntax 5-29
  record structure support 5-2
  removing a directory 5-14
Server FTP commands, summary 5-7
Server TELNET 6-18
  autologon to VTAM applications 6-19
  commands 6-18
    ACTEST 6-19
    BYE 6-18
    CLOSE 6-18
    END 6-18
    HELP 6-18
    LOGIN 6-18
    LOGOFF 6-18
    LOGOUT 6-19
    NETSTAT 6-18
    NEWS 6-18
    QUIT 6-18
    SIGNOFF 6-19
    SIGNON 6-18
  commands HELP + command name 6-18
  overview of 6-2
  USS Table support 6-19
show connections command 4-52
signature, in mail 7-5
Simple Mail Transfer Protocol (SMTP) 7-1
Simware Sim 3278 terminals 6-16
SITE command 4-46
  cancelling previous SUBMIT 5-24
  Client FTP 4-45
  Client FTP2 2-65
  Client FTP3 3-63
  parameter 5-17
    ATTR 5-17
    AUTOINDEX 5-17
    AUTOMOUNT 5-17
    AUTORECALL 5-17
    BLOCK 5-17
    CARDS 5-17, 5-20
    CD 5-17
    CHARSET 5-17
    CHKPTINT 5-17
    COMPACT 5-17
    CONDDISP 5-17
    CYLINDER 5-17
    DACLASS 5-18
    DATACLASS 5-18
    DATASETMODE 5-18
    DBCSSET 5-18
    DCBDSN 5-18
    DCLOSE 5-18
    DEVNULL 5-18
    DIDLE 5-18
    DIR 5-18
    DIRECTORYMODE 5-19
    DOPEN 5-19
    DSEQ 5-19
    DUMMY 5-19
    EXPDT 5-19
    filetype 5-20
    FORTRAN 5-17

fortran 5-20
FULLTRK 5-20
HALFTRK 5-20
HFS 5-20
IBUF 5-20
IRBLKSIZE 5-20
IRLRECL 5-20
IRRECFM 5-20
ISPFENQ 5-20
ISPRFES 5-20
JESLRECL 5-20
JESRECFM 5-20
LABEL 5-20
LINE 5-17, 5-21
LISTFMT 5-21
LKEDRES 5-21
LOADLIB 5-17, 5-20
LRECL 5-21
MANAGEMENTCLASS 5-21
MIGRATEVOL 5-21
MOUNT 5-21
MVS 5-20
NCP 5-21
NDAB 5-21
NLSTCASE 5-22
OBJECT 5-17, 5-20
OBUF 5-22
OVERWRITE 5-22
PAD 5-22
PARALLELMOUNT 5-22
PDSE 5-22
PERSIST 5-22
POP 5-23
PRIMARY 5-22
PRINT 5-17, 5-20, 5-22
PRIVATE 5-22
PUSH 5-23
QDISK 5-23
RDW 5-23
RECALL 5-23
RECFM 5-17, 5-23
REPLYFMT 5-23
RES 5-23
RESTART 5-24
RETPD 5-24
retpd 5-19
RLSEE 5-24
SECONDARY 5-24
SOURCE 5-17, 5-20, 5-24
SPACE 5-24
STCLASS 5-24
STORCLASS 5-24
STRIP 5-24
SUBMIT 5-24
TABS 5-24
TAPE 5-24
TERSE 5-24
TRACK 5-24
TRANOPT 5-25
UCNT 5-25
UMASK 5-25
UNIT 5-25
VBS 5-20, 5-25
VCNT 5-25
VERBOSE 5-25
VOLUME 5-25
VS 5-20, 5-25
VSEQ 5-25
WRAPRECORD 5-25
Server FTP 5-15
SMS storage class 5-24
SMTP 7-1
definition of 7-2
transporting electronic mail 7-2
SNDMSG
invoking 7-6
mail transport program 7-2
SNDS command, Client FTP2 2-66
software components 1-3
SOURCE parameter, SITE command 5-17, 5-20, 5-24
space allocation
extents 5-32
fragmented 5-32
in tracks 5-24
MVS 5-32
SPACE parameter, SITE command 5-24
specifying path names 4-30
split-screen operation, in Client TELNET 6-3
SPOOL#4, mail transport program 7-2, 7-9
SSID invocation option, Client FTP3 3-5
SSMTP, mail transport program 7-2, 7-8
STAT command

Client FTP 4-47
Server FTP 5-27
STATUS command
Client FTP2 2-67
Client FTP3 3-64
STCLASS parameter, SITE command 5-24
STOR, data transfer option 5-39
storage class 5-24
STORCLAS parameter, SITE command 5-24
STORCLASS parameter of FTP.DATA 3-16
stream mode 3-49, 4-33
in the MODE command 2-48, 3-49
STRIP parameter, SITE command 5-24
stripping pad characters 5-24
STRU command 5-42
Client FTP 4-48
STRUCT command
Client FTP2 2-68
Client FTP3 3-65
SUBMIT parameter SITE command 5-24
SUNIQUE command
Client FTP2 2-69
Client FTP3 3-65
supported ASCII terminal types 6-16
switching host prefixes 4-51
syntax conventions 2-12
syntax, FTP path name 5-29
SYS invocation option 2-9
Client FTP 4-10
SYSOPT for application selection 11-9
SYSPRINT file, allocating to terminal 6-4
SYSREQ key, on 3278 terminals 6-16
SYSTEM command, Client FTP3 3-66

T

tab
characters 5-40
horizontal 5-40
stops 5-24
vertical 5-40
TABS parameter, SITE command 5-24
tape data sets, on remote hosts 5-6
TAPE parameter, SITE command 5-24
TCP
checksums for end-to-end reliability 1-4
definition of 1-4
flow control on connections 1-4
full-duplex connections 1-4
invocation option, Client FTP3 3-5
segments 1-4
TCPIP.DATA 3-9, 3-11, 3-13
TCPIPJOBNAME parameter of TCPIP.DATA 3-13
TCPREXEC program 9-2
TELNET
3278 terminal 6-3
command
APPLID 6-4
invoking, See Client TELNET
SYS option 6-4
escape sequences 6-20
file structure 5-44
LINE option 6-4
TSO Client TELNET command 6-3
Virtual Line Terminal (VLT) facility 6-4
Terminal Monitor Program (TMP) 2-6, 3-7, 4-7
TERSE parameter, SITE command 5-24
TEST debugging option, See debugging
TEST1 debugging option, See debugging
testing control connections 2-12
3767 terminal 6-3
3278 terminal 6-3
throughput 3-2
TMP 11-11
TRACE command, Client FTP3 3-67
TRACE invocation option, Client FTP3 3-5
TRACK parameter, SITE command 5-24
TRANOPT parameter, SITE command 5-25
transferring data 3-2
transferring files 4-36, 4-44
in a single JCL job 2-79, 3-19, 4-58
to a host 5-3
transformation rules
binary files 5-49
character files 5-42
TRANSLATE invocation option, Client FTP3 3-5
transmission modes
block mode 4-33

compressed mode 4-33
stream mode 4-33
transmit control command, in TELNET 6-9
transmit escape command, in TELNET 6-9
transporting electronic mail 7-2
troubleshooting Client FTP 4-10
DISP option 4-10
TEST 4-11
truncating/folding records 5-39
TSO
FTP3 command 3-4
invoking Client FTP3 3-4
TSO CALL command 3-6
Client FTP 4-5
Client FTP2 2-6
TSO Client TELNET
network interface command, TELNET 6-3
PL/I Transient Library 6-3
program screen mode 6-3
TSO command, Client FTP3 3-68
TTCP TSO command processor 10-5
TTY, TSO Client TELNET command option 6-4
TYPE command
Client FTP2 2-70
Client FTP3 3-69

U

UCNT parameter SITE command 5-25
UDP, definition 1-4
UMASK parameter, SITE command 5-25
UNIT parameter, SITE command 5-25
UNITNAME parameter of FTP.DATA 3-16
UNIX
/etc/printcap file 8-3
print commands 8-3
print facility 8-3
remote printer definition 8-3
using
Client FTP 4-12
Client FTP2 2-11
Client FTP3 3-17
Client TELNET 6-3
USMTP
domain literal 7-12
file inclusion 7-11
host name mapping 7-11
mail transport program 7-2, 7-11
MX record support 7-11
USPOOL
backspacing 8-8
carriage control 8-9
configuration 8-9
control characters, translation 8-8
control file format 8-6
data file format 8-6
definition of 8-2
horizontal tab 8-8
invoking 8-8
line feeds 8-9
lpr protocol 8-4
print server facility 8-8
USS Table support for Server TELNET 6-19
USSTAB 6-19

V

VBS parameter, SITE command 5-20, 5-25
VCNT parameter, SITE command 5-25
VERBOSE invocation option, Client FTP3 3-5
VERBOSE parameter, SITE command 5-25
Virtual Terminal Facility (VTF), for TELNET 6-18
VLT debugging option
Client FTP 4-11
Client FTP2 2-10
VLT facility, in TELNET 6-4
VOLUME parameter of FTP.DATA 3-16
VOLUME parameter, SITE command 5-25
volume table of contents (VTOC) 5-32
volumes
maximum number for output data set 5-25
specifying 5-25
VS parameter, SITE command 5-20, 5-25
VSEQ parameter, SITE command 5-25
VTAM Client TELNET
3278 terminals, full-duplex local echo mode 6-15

control characters 6-15
full-screen operation 6-16
invoking 6-14
NVT operation from 3278 terminals 6-15
operation 6-14
VTAM, specifying applications 1-6
VTOC, volume table of contents 5-32

W

WAIT invocation option
Client FTP 4-10
Client FTP2 2-10
WHOIS TSO command processor 10-8
WRAPRECORD parameter of FTP.DATA 3-16
WRAPRECORD parameter, SITE command 5-25
writing multiple data sets to tape 5-5